E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Diseño y Test de Convertidores Analógico – Digitales no lineales configurables usando Arduino Due

Grado en Ingeniería
en Tecnologías de Telecomunicación

Trabajo Fin de Gado

Autor: Javier Beloso Legarra

Tutor: Dr. Carlos A. De La Cruz Blas

Pamplona, 28 de Junio de 2016

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

A mi padre

RESUMEN

En el presente proyecto fin de grado se va a diseñar, implementar y testear una plataforma flexible y configurable de Conversores Analógico Digitales (CAD) no lineales. La plataforma consiste de dos partes; la primera está formada por un circuito integrado diseñado en la UPNA que contiene la parte analógica del CAD. La segunda es un microcontrolador basado en Arduino, que será la encargada de controlar la parte analógica actuando como parte digital del sistema y generando los algoritmos de conversión. La integración de ambas partes se hará a través de un diseño y fabricación de un PCB a modo de shield que se va a acoplar a la tarjeta Arduino.

Palabras clave: Conversor Analógico Digital, Arduino, Algoritmo, Shield, Linealizar.

ABSTRACT

In this current final degree project, an adaptable and configurable platform of non-linear Analog to Digital Converters (ADC) will be designed, implemented and tested. The platform is based on two parts; the first one is formed by an integrated circuit which has been designed in the Public University of Navarra (UPNA), and it contains the main analog building blocks of the ADC. The second part is a microcontroller that is based on Arduino, and it will control the analog part performing the digital processing of the system and providing the proper conversion algorithms. The hardware integration of the two parts will be achieved by the design and manufacture of a PCB board acting like a shield that will be connected to Arduino.

Key words: Analog to Digital Converter, Arduino, Algorithm, Shield, Linearize.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ABREVIATURAS

CAD: Conversor Analógico Digital

CDA: Conversor Digital Analógico

CISC: Complex Instruction Set Computer

DIP: Dual In – line Package

DRC: Design Rules Check

ESR: Equivalent Series Resistor

IDE: Integrated Development Environment

PCB: Print Circuit Board

PWM: Pulse Width Modulation

RISC: Reduced Instruction Set Computer

SAR: Successive Approximation Register

SMD: Surface Mount Device

SNR: Signal to Noise Ratio

SOT: Small – Outline Package

S&H: Sample & Hold

UART: Universal Asynchronous Receiver – Transmitter

1. INTRODUCCIÓN

1.1. MOTIVACIÓN DEL PROYECTO

En los últimos años se ha podido comprobar que la tendencia de desarrollo en la industria electrónica gira en torno al mundo digital, dejando de lado poco a poco al mundo analógico. Las ventajas que posee la tecnología digital son evidentes con respecto a la analógica, por ejemplo el rápido crecimiento tecnológico, la fácil integración entre sistemas, el almacenamiento de la información y el tratamiento de esta o la mayor robustez frente al ruido. Pero el mundo que nos rodea y el tipo de señales que nos proporciona son de naturaleza analógica, por ese motivo aunque la mayor parte de un sistema sea de carácter digital, nunca podrá desaparecer esa pequeña parte analógica. Por ese motivo es necesario un elemento que sirva de interfaz entre ambos mundos: el **conversor analógico – digital**, **conversor A/D** o **CAD**.

Generalmente la señal de entrada será de naturaleza analógica y en la mayoría de los casos provendrá de un elemento sensor o de algún tipo de transductor de señal. Cabe remarcar que este tipo de señales suelen ser bastante ruidosas y de una amplitud bastante pequeña, y además pueden relacionarse de manera no lineal con la magnitud a medir. Por ese motivo suele requerirse un sistema de acondicionamiento de señal que se encargue de "limpiar" la señal y dejarla preparada para un futuro procesado. En la parte de acondicionamiento se incluyen amplificadores, filtros, etc. encargados de adaptar la señal al conversor. También puede encargarse de linealizar la señal procedente del sensor en caso de que sea de naturaleza no lineal, presentando una función de transferencia inversa a la del sensor. En la siguiente figura se muestra un esquema típico de un sistema de instrumentación incluyendo la parte de linealización en el acondicionamiento de la señal.

Figura 1: Sistema de instrumentación con linealización en la parte de acondicionamiento

Otro método alternativo, el cual va a ser usado en el presente proyecto fin de grado, es la linealización empleando el mismo CAD, lo que se conoce como **CAD no lineal**. El sistema tendrá una doble finalidad; por un lado se encargará de digitalizar la señal de entrada, y por otro lado se encargará de linealizarla. A modo de ejemplo, supóngase una señal de entrada al CAD de tipo sinusoidal. A la salida del mismo se tendrá dicha señal linealizada y digitalizada. En la siguiente figura se muestra mejor este concepto.

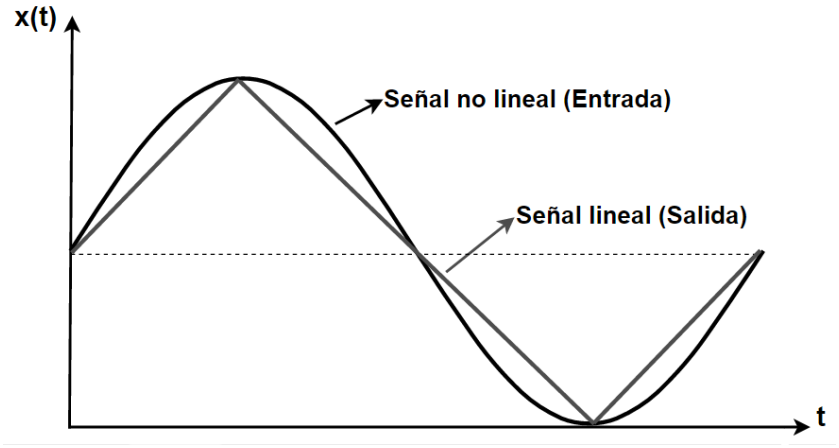


Figura 2: Proceso de linealización de la señal de entrada

De este modo la linealización y la conversión se realizan en el mismo dispositivo optimizando y flexibilizando el procesado y simplificando la parte de acondicionamiento del sistema de instrumentación. En la siguiente figura se muestra la idea. Aunque aparezca la parte de acondicionamiento, en el proyecto sólo se va a desarrollar el conversor.

Figura 3: Sistema de instrumentación con linealización en la parte de conversión A/D

El objetivo principal del presente proyecto fin de grado es crear una plataforma flexible, programable y de propósito general del CAD no lineal empleando un circuito integrado analógico fijo ya fabricado y un microcontrolador basado en **Arduino** para configurarlo. El integrado posee la parte analógica del conversor, y mediante Arduino se controlará a este, actuando como parte digital del sistema y generando los algoritmos de conversión.

Arduino es una plataforma de hardware libre basada en una placa con un microcontrolador Atmel®, la cual proporciona toda la electrónica necesaria para ser configurada mediante un entorno de desarrollo, IDE. Existen múltiples modelos de Arduino en el mercado, de los cuales unos son más potentes que otros, o están destinados a diferentes aplicaciones El modelo de Arduino que se va a emplear es el "**Arduino Due**", el cual es uno de los más potentes del mercado en cuanto a características se refiere. De esta manera se consigue reducir el tiempo de desarrollo que requeriría una plataforma de testeo con un microcontrolador convencional, en la que habría que incorporar además del microcontrolador, su circuitería asociada, el propio sistema a controlar así como diversos interfaces de comunicación.

1.2. OBJETIVOS

Con los dos elementos anteriores, se tienen los ingredientes principales para presentar los objetivos del proyecto.

1. Elaboración de una **plataforma** de implementación y testeo compacta y altamente configurable de un CAD no lineal, apoyándose en el sistema Arduino para facilitar su desarrollo. Dicha plataforma debe servir de motivación para el desarrollo de futuras plataformas adaptadas al sistema a controlar, bien sea un CAD o cualquier otro sistema.
2. Creación de un PCB con la parte analógica del CAD para crear un **shield** en la placa Arduino. Un shield se trata de un módulo que posee una funcionalidad determinada el cual se puede acoplar físicamente a la placa Arduino mediante los diferentes puertos que esta posee.
3. La parte analógica del CAD sigue una arquitectura de **integrador de doble rampa**, la cual es adecuada para los propósitos del proyecto por sus bloques modulares y fácil diseño de las partes y su configurabilidad.
4. A través de la **programación del microcontrolador** se controlarán unos puertos determinados de la parte analógica del conversor, configurando la frecuencia de las señales, las tensiones de referencia, el reseteo de los integradores, así como la secuencia de activación de los diferentes bloques analógicos.
5. Se realizarán diferentes **algoritmos de conversión** con el fin de probar el buen funcionamiento de la plataforma en diferentes escenarios y obtener los principales parámetros del conversor.

1.3. ORGANIZACIÓN DE LA MEMORIA

A lo largo de la memoria se irá explicando cómo ha sido el proceso de desarrollo de la plataforma. En el capítulo 2 se hará una introducción a los CAD, en la que se comentará como es todo el proceso de conversión de la señal analógica a digital, cuales son las características de un conversor, sus errores más comunes así como las arquitecturas típicas de los diferentes tipos de conversores. En el capítulo 3 se analizará de forma global como va a ser la plataforma, realizando un estudio sobre el funcionamiento del integrado así como las modificaciones que ha sufrido con respecto a los esquemas de la bibliografía habitual. También se estudiará la plataforma Arduino, analizando sus características más importantes tales como sus puertos, sus características eléctricas o lo timers, los cuales serán el elemento clave en la programación de los algoritmos. Una vez se tenga una perspectiva global de la plataforma, se detallará todo el proceso de elaboración de la misma en el capítulo 4, como la problemática de alimentación del integrado, la elaboración del PCB o los diferentes algoritmos de conversión. Los resultados preliminares obtenidos se comentarán en el capítulo 5 y las conclusiones del proyecto en el capítulo 6. Finalmente se comentarán las futuras líneas de desarrollo que aporten mejoras a la plataforma en el mismo capítulo 6.

1.4. AGRADECIMIENTOS

Me gustaría agradecer toda la ayuda y el apoyo prestado por mi tutor de trabajo de fin de grado Carlos A. De La Cruz Blas, quien siempre me ha echado una mano cuando se lo he pedido y me ha sacado de algunos apuros cuando no me salían las cosas. También por haberme permitido colaborar en una asignatura para la elaboración de una nueva práctica, lo cual fue una actividad muy interesante para mí y que sirvió para hacer algo que todos lo pudieran emplear. También me gustaría agradecer todo el apoyo prestado por mis padres, mis tíos, mis amigos y mis compañeros de clase ya que sin ellos, muchas veces habría resultado muy duro seguir para adelante. También por darte motivación y hacerte creer en ti mismo, que muchas veces son aspectos que se nos olvidan. Doy un saludo a mis compañeros de laboratorio así como el chico de intercambio de México con quienes he podido charlar y conversar así como aprender nuevas cosas. Por último, decir que han sido cuatro años inolvidables, llenos de nuevas experiencias, conocimientos, subidas y bajadas y muchos momentos que perdurarán en el recuerdo para toda la vida.

Finalmente, me gustaría dedicar este proyecto a mi padre, quien me ha demostrado que hasta el último momento nunca hay que perder la esperanza y las ganas de vivir.

2. CONVERSORES ANALÓGICO – DIGITALES

En el capítulo introductorio de la memoria, se ha citado la necesidad de un sistema que sirva de interfaz entre el mundo analógico y el mundo digital, conocido como conversor analógico – digital. Antes de explicar cómo ha sido el proceso de desarrollo de la plataforma de implementación y testeo del CAD, dicho dispositivo requiere un pequeño estudio detallado, para que nos vayamos familiarizando con su funcionamiento, sus parámetros más relevantes y conozcamos los tipos de conversores de forma breve para poder establecer una comparación entre ellos y seleccionar el más adecuado para la aplicación a desarrollar.

Empleando una definición formal, podría decirse que "*un conversor analógico – digital es un dispositivo electrónico que establece una relación biunívoca entre el valor de la señal en su entrada y la palabra digital que se obtiene a su salida*". En definitiva, un CAD transforma un valor analógico a un lenguaje digital, usado en el procesamiento de la información, computación, transmisión de datos y sistemas de control. En la figura 4 se muestra el esquema clásico de un CAD, relacionando la entrada analógica con la salida digital. Como se puede observar, la señal de entrada analógica debe estar comprendida en un determinado rango de valores que el conversor puede manejar. La señal V_{REF} suele ser un nivel de continua empleado como referencia para establecer la comparación y finalmente la salida es la palabra digital correspondiente a ese valor analógico.

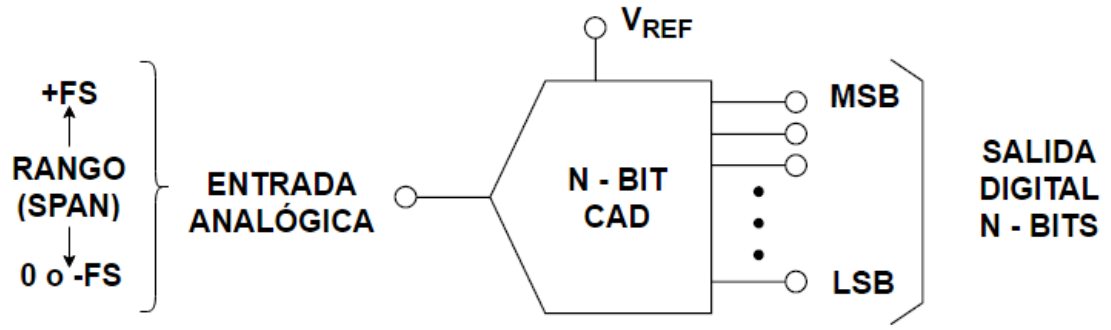


Figura 4: Esquema de un CAD

Dependiendo de la polaridad de la señal de entrada, existen dos tipos de conversores: los unipolares, y los bipolares. En los conversores **unipolares**, la señal de entrada posee únicamente valores positivos. Por el contrario, en los conversores **bipolares** la señal de entrada tiene valores positivos y negativos. Los conversores unipolares son más sencillos, pero los conversores bipolares son más útiles debido a que las señales suelen poseer polaridades positivas y negativas. En la siguiente figura se muestra la diferencia entre ambos conversores.

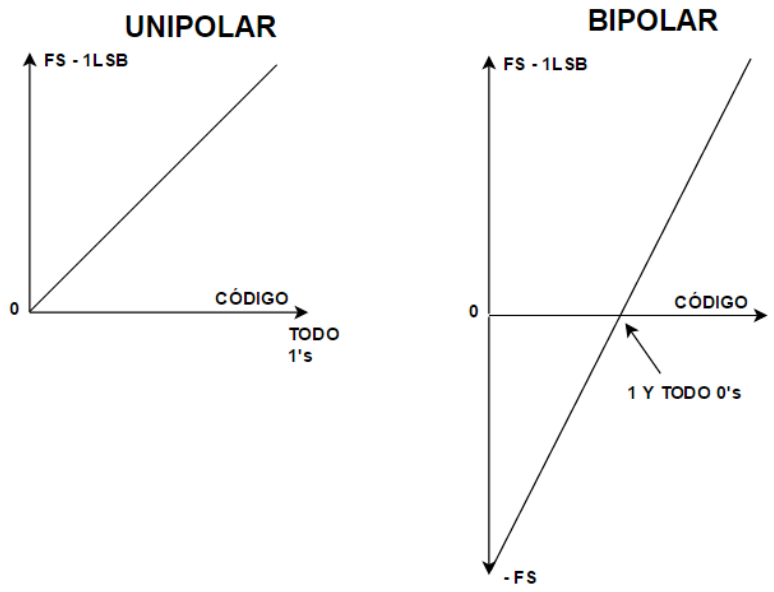


Figura 5: Conversor unipolar y bipolar

También se pueden encontrar conversores de tipo ***fully – diferential***. Este tipo de conversores se caracterizan porque la señal de entrada es de tipo diferencial. En audio, se les suele llamar también conversores balanceados. La ventaja de este tipo de circuitos es que son muy robustos frente al ruido que se acoplan a las señales así como la reducción de los armónicos de cualquier orden. Además son ideales para sistemas de bajo voltaje. El conversor que se va a emplear en el proyecto va a ser de tipo fully – differential, cuya señal de entrada va a ser diferencial. En la siguiente figura se muestra la diferencia entre un amplificador estándar y otro fully – differential. Se ha decidido mostrar un amplificador ya que ilustra muy bien la idea de entrada y salida diferencial. En el amplificador de la izquierda, la entrada y salida son diferenciales, mientras que en la derecha la entrada es diferencial pero la salida es unipolar (*single – ended*).

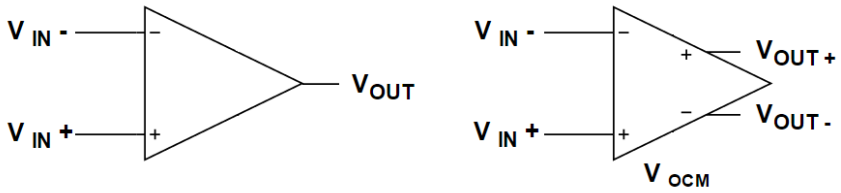


Figura 6: Amplificador "Single – ended" (izquierda) y amplificador "Fully – diferential" (derecha)

2.1. CONCEPTOS BÁSICOS

Conceptualmente, la conversión de una señal analógica a una señal digital se realiza en tres pasos: **muestreo**, **cuantificación** y **codificación**.

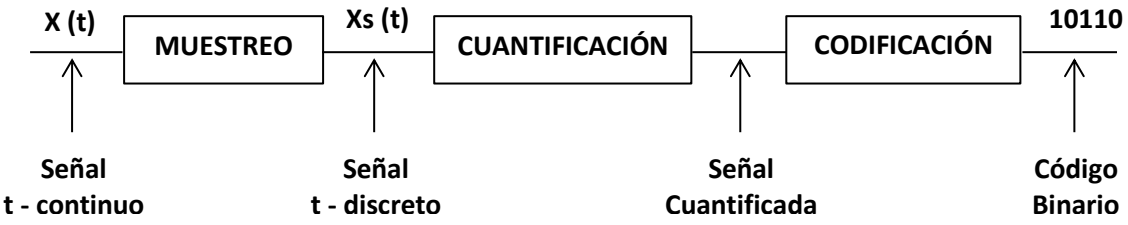


Figura 7: Proceso de conversión de una señal analógica a digital

*El **muestreo** es la transformación de tiempo continuo a tiempo discreto de la señal analógica mediante la extracción de muestras equiespaciadas de la señal de entrada, separadas mediante un periodo de muestreo Ts.* Se realiza la discretización del eje tiempo de la señal analógica. Siguiendo el teorema de Nyquist, el cual establece que la frecuencia de muestreo debe ser como mínimo el doble del ancho de banda de la señal, es posible recuperar la señal original a partir de esas muestras equiespaciadas sin pérdida de información. En la figura 8 se muestra un ejemplo de circuito de muestreo y retención S&H, y en la figura 9 el proceso de muestreo de una señal analógica.

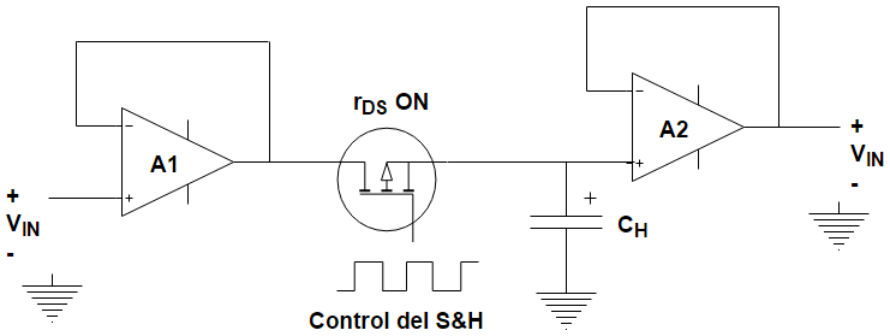


Figura 8: Arquitectura típica de un S&H

Figura 9: Proceso de muestreo de una señal analógica

Para no perder el espíritu matemático, se puede hacer una definición formal que explique el proceso de muestreo basándonos en la teoría del procesado de señal. Dada una señal a muestrear, $x(t)$ y un tren de deltas que representen el instante en el que tomamos la muestra, $s(t) = \sum_{-\infty}^{\infty} \delta(t - nT_s)\,; n \in \mathbb{Z}$ separadas con un determinado periodo de muestreo T_S, se tiene que al multiplicar ambas señales obtenemos la señal muestreada $x_S(t) = x(t)s(t) = \sum_{-\infty}^{\infty} x(nT_S)\delta(t - nT_s)$.

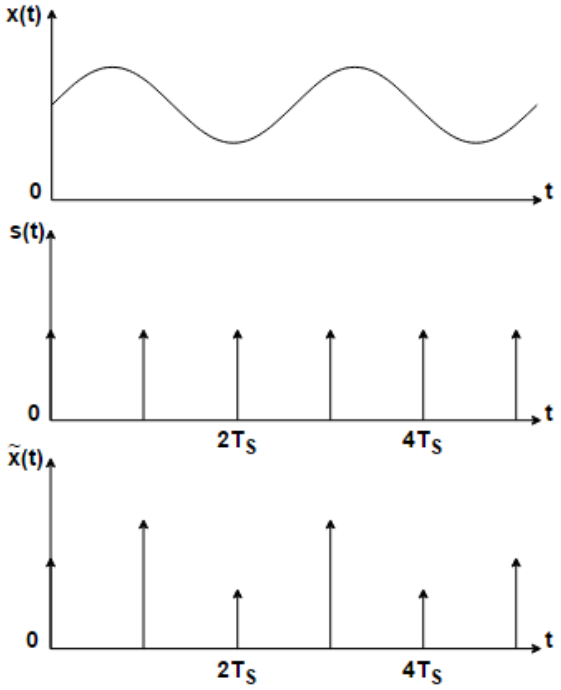


Figura 10: Proceso de muestreo de una señal analógica en el dominio temporal, donde se relaciona la señal a muestrear con sus instantes de muestreo obteniendo la señal discreta

Como siempre, cualquier acción que se lleve en el dominio temporal, implica un cambio en el dominio frecuencial. De esta manera, dado el espectro de la señal a muestrear, $X(f)$ y el espectro de un tren de deltas que representen el instante en el que tomamos la muestra, $S(f) = \frac{1}{T_S}\sum_{-\infty}^{\infty} \delta(f - kf_s)\,; k \in \mathbb{Z}$ separadas con una determinada frecuencia de muestreo f_S, se tiene que al convolucionar ambas señales obtenemos el espectro de la señal muestreada $X_S(f) = X(f) * S(f) = \frac{1}{T_S}\sum_{-\infty}^{\infty} X(f - kf_s); k \in \mathbb{Z}$.

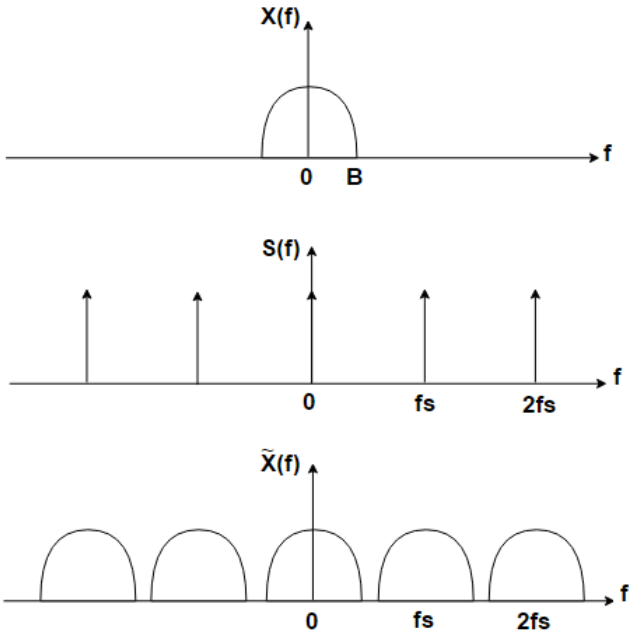


Figura 11: Proceso de muestreo de una señal analógica en el dominio frecuencial, donde se relaciona el espectro de la señal con sus instantes de muestreo, obteniendo la periodización de los espectros

Como se puede observar en las figuras 10 y 11, el muestreo de una señal, implica la periodización de los espectros.

Como se ha citado anteriormente, el criterio de Nyquist establece que para una señal limitada en banda, la frecuencia de muestreo mínima debe ser el doble del ancho de banda de la señal, es decir, se debe cumplir que:

$$|X(f)| = 0, \forall f \geq B$$

$$f_S \geq 2BW$$

Se denomina **tasa de Nyquist** al valor mínimo $f_S = 2BW$. Al intervalo de frecuencias centrado en cero con límites en $\pm f_S/2$, se le llama **intervalo de Nyquist**.

Este criterio tiene mucho sentido y es muy importante tenerlo en cuenta, ya que como se ha observado en la figura 11, al realizar el muestreo de la señal, los espectros tienden a periodizarse a intervalos de la frecuencia de muestreo f_S. Por ese motivo, si no se cumple dicho criterio y tenemos una frecuencia de muestreo que esté por debajo, al desplazarse los espectros puede que se solapen, dando lugar a un efecto conocido como **aliasing**, el cual distorsiona la información. Por ese motivo, en todo sistema que emplee un CAD se debe colocar un **filtro antialiasing** de paso bajo cuyo ancho de banda máximo sea $f_S/2$, y que limiten en banda a las señales de entrada, con el fin de evitar este efecto.

De forma general, se establece que una señal con un ancho de banda BW debe ser muestreada a una velocidad que sea como mínimo el doble del ancho de banda de la señal, esto es $f_S \geq 2BW$ para no perder y distorsionar la información. Si no se cumple el criterio y

tenemos que fs < 2BW, fenómeno conocido aliasing, se colarán replicas espectrales en el intervalo de Nyquist fs/2 llamadas alias, a frecuencias f_S – BW.

*La **cuantificación** es el proceso en el cual el valor de cada muestra se sustituye por el valor más próximo de un conjunto finito de valores accesibles, llamados niveles de cuantificación*. En este proceso se realiza la discretización del eje que incorpora la información acerca de la amplitud de las muestras. En definitiva, para infinitos valores de entrada, tendremos un conjunto finito de valores a la salida. Dado que se está realizando una aproximación a un valor, supone un proceso con pérdida irreversible de información. Para una conversión lineal, el tipo de cuantificación más común es la uniforme en el que los niveles son todos equiespaciados. Para conseguir una función de transferencia distinta de la lineal la cuantificación que emplearemos será la no uniforme.

En la figura 12 se muestra un ejemplo de cuantificación, correspondiente a la **función de transferencia** de un CAD de 3 bits, la cual relaciona la salida digital con la entrada analógica. En el eje de abscisas se tiene la señal analógica y en el eje de ordenadas la señal o palabra cuantificada. Conceptualmente, es un proceso bastante simple. Lo que se hace es aproximar ese rango de valores analógicos a un valor cuantificado.

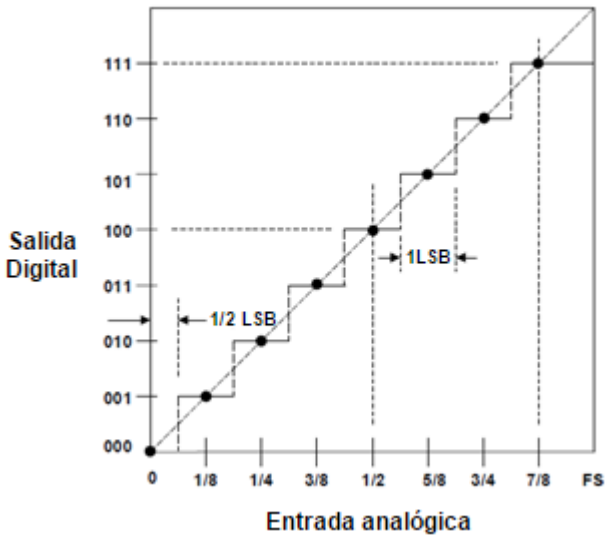


Figura 12: Función de transferencia de un CAD de 3 bits

Existen dos tipos de cuantificación, que dependen de la forma en la que se asigne el valor cuantificado al valor de entrada. Los niveles de decisión se colocan en el eje de abscisas y los niveles de cuantificación en el eje de ordenadas.

- **Cuantificación por redondeo**: cuando a una amplitud de entrada nula se le asigna se le asigna un nivel de cuantificación.

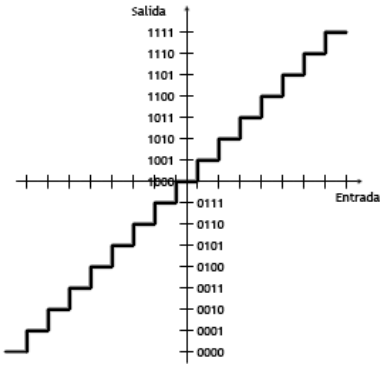


Figura 13: Cuantificación por redondeo

- **Cuantificación por truncamiento**: cuando a una amplitud de entrada nula se le asigna se le asigna un nivel de decisión.

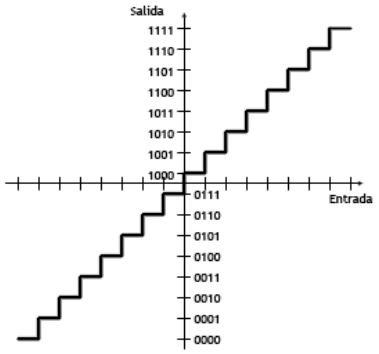


Figura 14: Cuantificación por truncamiento

Adicionalmente, también existen dos tipos de cuantificación, dependiendo de cómo sea la distribución de los escalones de cuantificación.

- **Cuantificación uniforme**: En los cuantificadores uniformes (cuantificación lineal) la distancia entre los niveles de cuantificación es siempre la misma. No hacen ninguna suposición acerca de la naturaleza de la señal a cuantificar, por lo que trata a todas las muestras por igual.
- **Cuantificación no uniforme**: La cuantificación no uniforme (cuantificación no lineal) se aplica cuando se procesan señales no homogéneas que se sabe que van a ser más sensibles en una determinada banda concreta de frecuencias. En este caso, lo que se hace es estudiar la propia entropía de la señal y asignar niveles de cuantificación de manera no uniforme, de tal modo que se asigne un mayor número de niveles para aquellos márgenes en que la amplitud cambia más rápidamente (contienen mayor densidad de información).

Para terminar con el concepto de cuantificación, hay que remarcar que en ese proceso de aproximación, se produce siempre un error, conocido como **error de cuantificación**. Básicamente, el error de cuantificación se define como la diferencia entre el valor de la muestra y su valor cuantificado. En la figura 15 se muestra la función de transferencia de un CAD así como la curva del error. Mediante un adecuado tratamiento matemático – estadístico de ese error, se modelará el error de cuantificación como un ruido blanco aditivo, permitiendo establecer una relación de SNR entre la potencia de la señal de entrada al cuantificador y la potencia de ese error de cuantificación. El objetivo de este procedimiento es caracterizar la calidad del CAD, de tal manera que cuanta mayor SNR tenga el conversor, menor error de cuantificación introducirá y por lo tanto más preciso será. Un buen criterio lógico podría ser que a mayor número de bits de cuantificación del conversor, menor será el error de cuantificación, y por lo tanto mejor será la SNR. Hoy en día existen técnicas de procesado muy interesantes que intentan maximizar la SNR reduciendo el número de bits, aunque dichas técnicas se salen del objetivo de dicho proyecto.

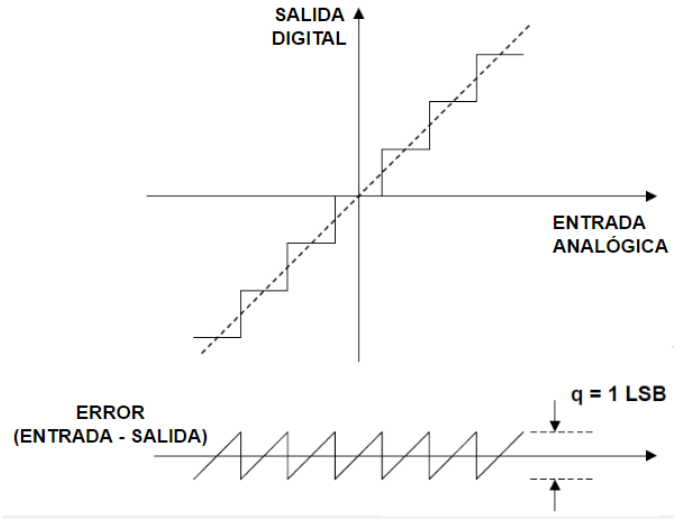


Figura 15: Relación entre la función de transferencia de un CAD y el error de cuantificación

En la siguiente figura, se muestra un diente de sierra que modela el análisis del ruido de cuantificación.

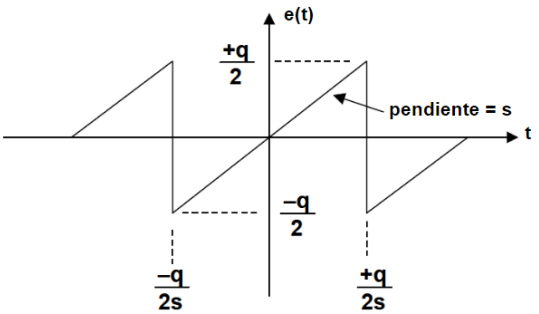


Figura 16: Modelado del error de cuantificación como un diente de sierra

A través de las siguientes ecuaciones se resumen algunos parámetros estadísticos de la señal de la figura 16 que se suelen emplear en el modelo del error de cuantificación.

$$ERROR = e(t) = st, \frac{-q}{2s} < t < \frac{+q}{2s}$$

$$ERROR\ MEDIO = \overline{e^2(t)} = \frac{s}{q}\int_{-q/2s}^{+q/2s} (st)^2 dt = \frac{q^2}{12}$$

$$ERROR\ CUADRÁTICO\ MEDIO = \sqrt{\overline{e^2(t)}} = \frac{q}{\sqrt{12}}$$

Finalmente, *la **codificación** es el proceso en el cual se asigna un código de b – bits a cada nivel de cuantificación, obteniendo a la salida una secuencia binaria.* El número de bits necesarios depende del número de niveles de cuantificación a ser codificados. Este paso hay que tenerlo siempre en cuenta ya que puede hacer que obtengamos datos erróneos, sobre todo cuando el sistema admite señales positivas y negativas con respecto a masa, momento en el cual la salida binaria del convertidor nos da tanto la magnitud como el signo de la tensión que ha sido medida.

Con estos tres pasos, se es capaz de digitalizar una señal analógica. A partir de aquí tendríamos un conjunto de bits correspondientes a la señal analógica, aptos para su procesado. Una vez que se han procesado y tratado dichos datos, es posible reconvertirlos a una señal analógica, mediante un conversor **digital – analógico** o **CDA**. En la figura 17 se muestra el esquema clásico de un CDA, relacionando la entrada digital con la salida analógica. Para ello se realiza el proceso inverso de la conversión analógico – digital: la **descodificación** y la **reconstrucción** o **interpolación**. En figura 18 se muestra dicho concepto. Cabe remarcar que no existe el paso de la "des cuantificación", ya que como hemos citado se trata de un proceso irreversible.

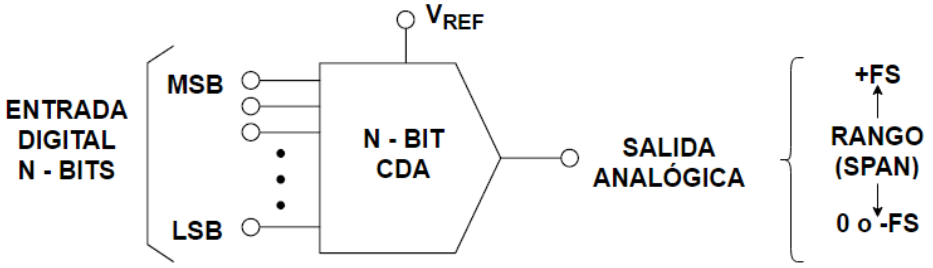


Figura 17: Esquema de un CDA

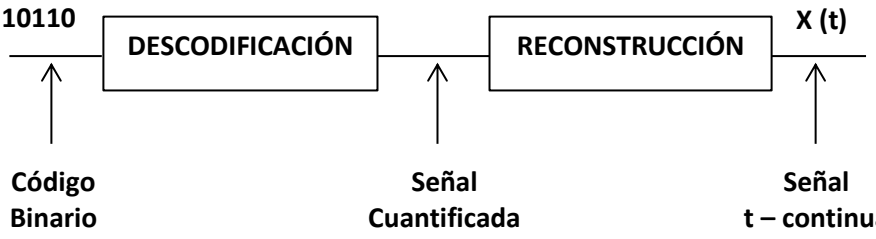


Figura 18: Proceso de conversión de una señal digital a analógica

2.2 CARACTERÍSTICAS DE UN CONVERSOR A/D

Un CAD, como cualquier sistema, presenta unas determinadas características que se deben tener en cuenta a la hora de emplearlo en un diseño determinado. Dependiendo de la aplicación a desarrollar, seleccionaremos un conversor con unas características u otras las cuales podrán consultarse en el **datasheet** del componente. A continuación se muestran las características de un CAD.

Precisión absoluta (*Accuaracy Absolute*): Para un código de salida dado, es la diferencia entre el voltaje de entrada analógica actual y el teórico requerido para producir el código.

Las fuentes de error suelen ser el error de ganancia, errores de linealidad y ruido.

Precisión relativa (*Accuaracy Relative*): Expresado en ppm (partes por millón), % o fracciones de 1 LSB (*Less Significate Bit*), es la desviación del valor analógico de cualquier código de su valor teórico después de haberse calibrado el rango de entrada (ver más adelante "Rango de entrada").

Tiempo de adquisición (*Acquisition Time*): Tiempo durante el cual el circuito de muestreo y retención debe permanecer en estado de muestreo para asegurar que el posterior estado de retención esté dentro de la banda de error especificada para la señal de entrada.

Aliasing (*Aliasing*): Serán las réplicas espectrales que se colarán en el intervalo de Nyquist si no cumplimos el criterio de muestreo. Dicho parámetro nos indicará de manera indirecta el ancho de banda máximo de la señal que podemos introducir al conversor, también conocido como **ancho de banda analógico** (*Analog Bandwidth*), ya que la tasa de muestreo es un parámetro fijo del sistema. Por lo tanto, el ancho banda máximo de la señal será:

$$BW_{max} = \frac{f_S}{2}$$

Tiempo de apertura (*Aperture Time*): Tiempo empleado por el circuito de muestreo y retención para conmutar de un estado de alta impedancia a impedancia cero, o lo que es lo mismo, de interruptor abierto o interruptor cerrado.

Inyección de carga (*Charge Injection*): Es un offset que se introduce en el condensador del circuito de muestreo y retención a través del conmutador y capacitancias parásitas. Esto puede dar lugar a errores en la conversión. El error de offset es proporcional a la carga:

$$Error\ de\ offset = Carga/Capacidad = \Delta Q/C$$

El error se puede reducir incrementando el condensador, pero esto implicará un incremento en el tiempo de adquisición (ver más adelante "Tiempo de adquisición").

Anchura de código (*Code Width*): El mínimo valor a la entrada que hace que se produzca a la salida un código digital.

Conversión completa (*Conversion Complete*): Es una salida que indica que ha finalizado la conversión de la señal. Se le suele conocer con el nombre de "fin de conversión" (*End O Conversion, EOC*).

Tiempo de conversión (*Conversion Time*): Tiempo empleado por el sistema para completar la conversión. Se mide como el transcurrido desde que el convertidor recibe una señal de "inicio de conversión" (*Start Of Conversion, SOC*), hasta que aparece un dato válido a la salida, la cual se indica con la salida EOC.

Velocidad de caída (*Droop Rate*): Cuando el circuito de muestreo y retención emplea un condensador para almacenar el dato, este no mantendrá la información de manera permanente. Por lo tanto, la velocidad de caída es la velocidad con la que cambia el voltaje de salida del condensador y pierde el dato.

Número efectivo de bits (*Efective Number of Bits*): Como se ha indicado en el apartado 2.1., mediante un adecuado tratamiento matemático – estadístico del error de cuantificación, puede establecerse una relación de SNR. Si introducimos una onda sinusoidal a la entrada del conversor y se realiza dicho tratamiento, se demuestra la siguiente relación:

$$SNR = 6.02N - 1.25\ (dB), con\ N = nº.\ de\ bits\ del\ conversor$$

De la cual se puede despejar el número de bits que emplea el conversor.

$$N = (SNR + 1.25)/6.02$$

Por lo tanto, el número efectivo de bits son los bits empleados en el conversor para alcanzar la SNR deseada.

Rango de entrada (*Full – Scale Range, FSR*): Es el rango de valores de la señal de entrada que admite el conversor. Para el caso unipolar, irá de 0 – V_{FSV}, y para el caso bipolar irá de $-V_{FSV}$ – V_{FSV}, siendo V_{FSV} la **tensión de fondo de escala** (*Full Scale Voltage, FSV*).

Glitch (*Glitch*): Son transitorios que se acoplan en forma de "picos" (glitches) de la señal digital (reloj o datos) a la salida analógica. Son muy rápidos, uniformes y difíciles de filtrar.

Distorsión armónica total (*Total Harmonic Distortion, THD*): Relación de la amplitud de los armónicos de la señal con la frecuencia fundamental de esta. Una señal será de mejor calidad cuanta menor distorsión armónica tenga.

$$THD = \frac{\sum Potencia\ de\ los\ armónicos}{Potencia\ de\ la\ frecuencia\ fundamental}$$

Impedancia de entrada (*Input Impedance*): Es la impedancia que presenta el conversor vista desde los bornes de entrada.

Linealidad (*Linearity*): El error de linealidad de un conversor es la desviación del valor analógico, en una gráfica que representa la relación de conversión, de la línea recta.

Sobrecarga (*Overload*): Valor que excede el rango de entrada del conversor produciendo una situación de sobrecarga.

Resolución (*Resolution*): El mínimo valor que puede distinguir el convertidor en su entrada analógica, o lo que es lo mismo, la mínima variación de la señal de entrada que se necesita para cambiar en un bit la salida digital.

Frecuencia de muestreo (*Sampling Rate*): El número de muestras por segundo que es capaz de adquirir el conversor.

Tiempo de estabilización (*Settling Time*): Tiempo entre la señal retenida y el definitivo asentamiento de la señal, dentro de la banda de error especificada.

Slew Rate (*Slew Rate*): Velocidad a la cual el valor de salida del circuito de muestreo y retención converge al valor muestreado deseado. Se mide en V/s.

Margen dinámico libre de espúereos (*Spurious Free Dinamic Range, SFDR*): Margen de variación de la potencia de entrada en la que la salida está libre de espurios.

2.3. ERRORES EN UN CONVERSOR A/D

Suele decirse que si el mundo fuera ideal, no existiría ingeniería. Todo sería perfecto y no existirían problemas que habría resolver con el fin de obtener nuestro objetivo. Podría decirse que en la ingeniería, cuando se realiza un diseño, se parte de un modelo cuasi – ideal, y al implementarlo surgen los problemas que se deben resolver. La R.A.E. (Real Academia de la lengua Española) define error, aplicada a la física y las matemáticas, como "*la diferencia entre el valor medido o calculado y el real*". En consecuencia, y dado que no vivimos en un mundo ideal, un conversor no es un sistema perfecto y presenta una serie de errores que se deben tener en cuenta. Un error es otro parámetro de un conversor, el cual también se puede consultar en un **datasheet**. A continuación se va a hacer un repaso de los errores más comunes en un CAD.

Error de ganancia (Gain Error): Produce un fondo de escala incorrecto. Un error de ganancia positivo provoca que un fondo de escala analógico se alcance con un código digital de todo 1's'. Un error de ganancia negativo hace que el valor digital de todo 1's' sea producido por un valor analógico menor que el fondo de escala.

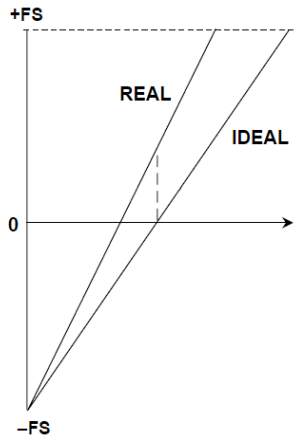


Figura 19: Error de ganancia

Error de linealidad (Linearity Error): Desviación entre la curva de salida teórica y la real, de modo que para iguales incrementos en la entrada, la salida indica distintos incrementos.

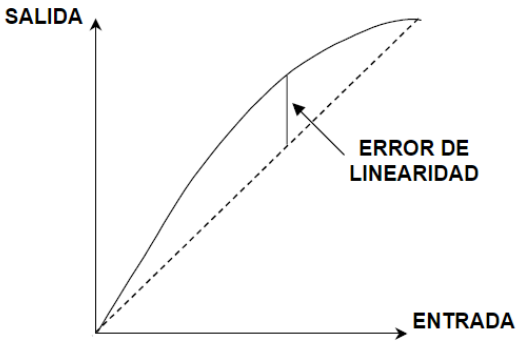


Figura 20: Error de linealidad

Error de monoticidad (Monotonic Error): Implica que la función de transferencia del conversor no es creciente. Ante incrementos de la señal de entrada, la salida puede decrecer.

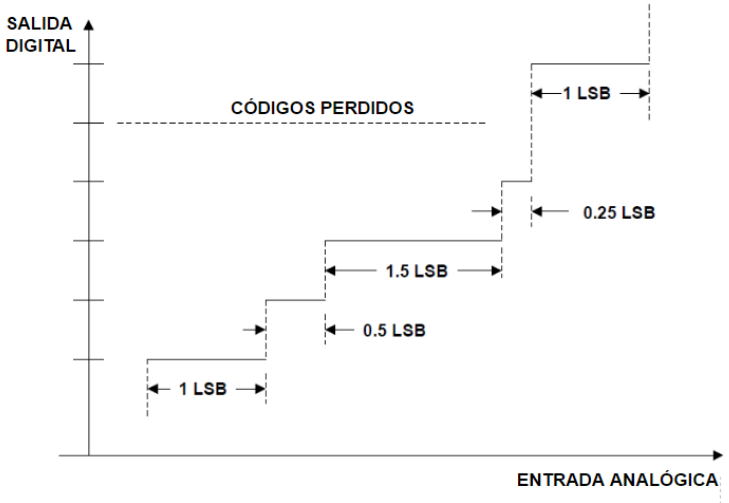


Figura 21: Error de monoticidad

Error de offset (Offset Error): Es un desplazamiento constante para todos los valores de la curva característica del conversor. Este error afecta a todos los códigos de salida por igual, y puede ser compensado por un proceso de ajuste.

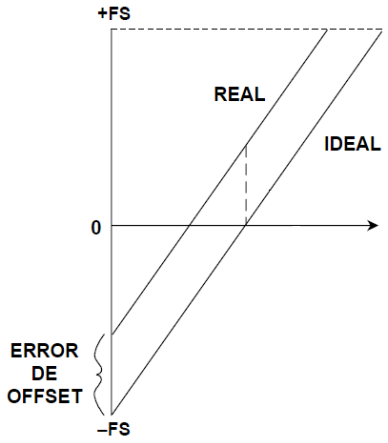


Figura 22: Error de offset

Error de cuantificación (*Quantization Error*): Diferencia entre el valor de la muestra y su valor cuantificado. Como se ha visto, en la figura 15 se muestra este tipo de error.

2.4. ARQUITECTURAS DE CONVERSIÓN A/D

A continuación se van a mostrar las arquitecturas de conversión analógico – digitales más comunes, con el fin de conocer diferentes estrategias de conversión. De forma general, y como ya hemos citado en el apartado 2 y en la figura 4, en un conversor la relación entre la salida digital y la entrada analógica depende de un valor de referencia, y la precisión de esta tensión de referencia condiciona la precisión del conversor. Esta tensión de referencia puede ser interna o externa al conversor. Normalmente esta tensión de referencia debe de estar dentro de un rango, cuyo valor máximo debe ser menor o igual a la tensión de alimentación del conversor V_{DD}.

2.4.1. COMPARADOR

Un comparador es el CAD más simple que podemos encontrar ya que se trata de un convertidor de 1 bit. Prácticamente todos los tipos de CAD tienen comparadores en sus arquitecturas de conversión, por ese motivo es importante nombrarlo. Cuando se incorporan a un circuito integrado, su diseño debe contemplar la resolución, velocidad, potencia disipada, tensión de desviación, corriente de polarización y área que ocupa el chip.

Está formado por un amplificador trabajando sin realimentación. Su salida es un nivel lógico que indica cuál de sus dos entradas está a un potencial mayor. Suelen incorporar de histéresis para evitar posibles cambios falsos debido a señales ruidosas.

En la siguiente figura se muestra un comparador así como su curva de histéresis. Cuando la tensión en la entrada no inversora es mayor que la inversora, la salida de este pasa a un

nivel alto. Por el contrario, cuando la entrada inversora es mayor que la entrada inversora la salida pasa a un nivel bajo. La curva de histéresis indica que las tensiones de transición de nivel alto a bajo así como de bajo a alto son diferentes.

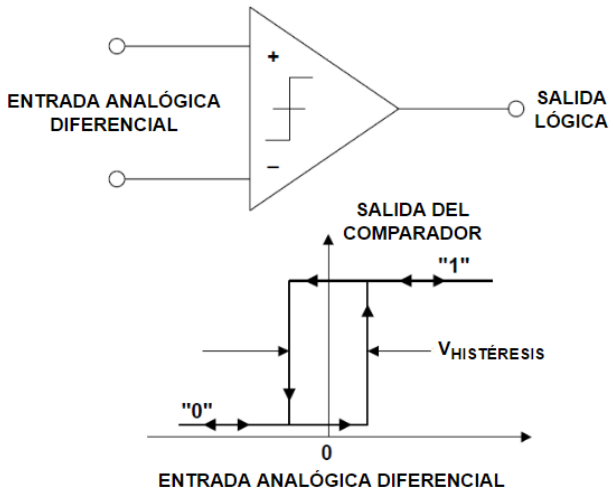


Figura 23: Comparador de 1 bit y su curva de histéresis

2.4.2. CAD PARALELO (FLASH)

El CAD Flash es el conversor más rápido, ya que la conversión se realiza de forma simultánea y casi instantánea por la velocidad que suelen alcanzar, pudiendo llegar hasta los cientos de MHz. La figura 24 muestra el esquema interno de un CAD de comparadores en paralelo de 3 bits. Está formado por una cadena de comparadores analógicos de alta velocidad y un codificador de prioridad. Cada comparador tiene una tensión de referencia que es 1 LSB mayor que el inmediato anterior en la cadena. Para un valor dado de la tensión de entrada, todos los comparadores por debajo de un cierto punto tendrán su entrada mayor que su referencia y en consecuencia aparecerá un "1" lógico en su salida. Todos los comparadores por encima de este punto, tendrán su referencia mayor que la entrada y aparecerá un "0" lógico a su salida.

El tiempo de conversión viene determinado por la velocidad de los compradores y el codificador. Entre el codificador de prioridad y los comparadores suele intercalarse un registro cuando la entrada varía rápidamente. El reloj que controla la transferencia de datos a través de los registros determina la velocidad de la salida.

El problema de este circuito es que se complica conforme aumenta el número de bits. En el esquema de la figura 24, para 3 bits de salida se requieren 7 comparadores. Es decir, para N bits se requieren 2^N-1 comparadores; en consecuencia, la adición de un bit duplica el número de comparadores. Además, al aumentar el número de bits también aumenta la complejidad del codificador de prioridad, ya que el número de puertas que requiere la lógica digital aumenta con el número de comparadores en un orden de N log (N), siendo N el número de comparadores. En consecuencia, estos modelos se emplean sólo en aplicaciones que requieran alta velocidad.

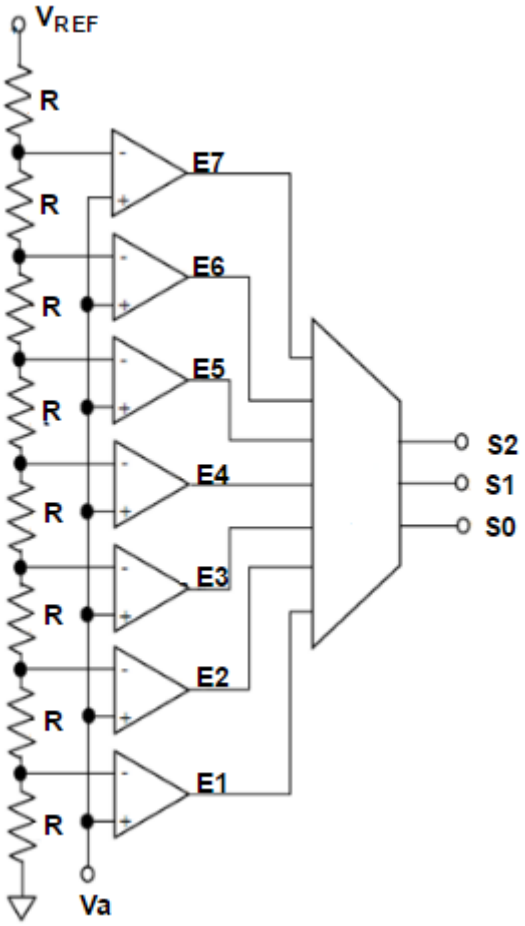


Figura 24: Conversor Flash

Esta configuración suele emplearse para la linealización de transductores, empleando la característica estática del CAD.

La tabla de verdad para el codificador de prioridad de CAD de la figura 24 se muestra en la tabla 1.

E7	E6	E5	E4	E3	E2	E1	S2	S1	S0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	1	0	1	0
0	0	0	0	1	1	1	0	1	1
0	0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1

Tabla 1: Tabla de verdad del codificador de prioridad de 7 entradas y 3 salidas del CAD de tipo Flash

La operación de este comparador es sencilla de mostrar mediante un ejemplo.

Supongamos: Vref=8 V y Va=3,5 V. Entonces E1= E2= E3=1 (nivel lógico), con el resto a cero. Entonces, según la tabla 2, S0=S2=1 y S1=0, que corresponde al código binario del número decimal 3.

2.4.3. CAD DE SEGUIMIENTO

También llamados de tipo "tracking", tienen una respuesta más rápida que el de conteo. En la figura 25 se muestra el esquema interno del convertidor. En ella se observa el contador ascendente – descendente, que determina la aproximación digital de la tensión analógica de entrada. Al principio se pone el contador a cero. El contador se incrementa según le llegan impulsos de reloj. La cuenta digital se va convirtiendo en analógica en el CDA y es comparada con la entrada. Mientras el resultado de la conversión D/A sea menor que la entrada, el comparador ofrece salida de nivel alto y continúa la cuenta ascendente ("Up"). Cuando la salida del CDA supera a la entrada, la salida del comparador pasa a nivel bajo, la cuenta disminuye en una unidad ("Down"). Ahora la salida del comparador será otra vez un nivel alto, la cuenta aumenta una unidad, la salida del CDA supera a la entrada y, así sucesivamente.

Una vez la salida del CDA haya alcanzado a la entrada, cualquier pequeño cambio que se produzca en ésta es seguido con rapidez por el circuito, contando o descontando. Como en estas situaciones se produce un seguimiento ("tracking") de la entrada, no hace falta introducir como etapa previa un circuito de muestreo y retención, S&H.

El tiempo de conversión aumenta proporcionalmente al número de cuentas. Es decir, existe un compromiso entre resolución y rapidez. Sin embargo, para pequeñas variaciones en la entrada, el circuito es rápido. La máxima velocidad de la señal de entrada que puede seguir el circuito (SR; Slew Rate) viene limitada por el periodo del reloj (Tclk) y responde a la siguiente expresión:

$$SR = \frac{1\ LSB}{T_{CLK}}$$

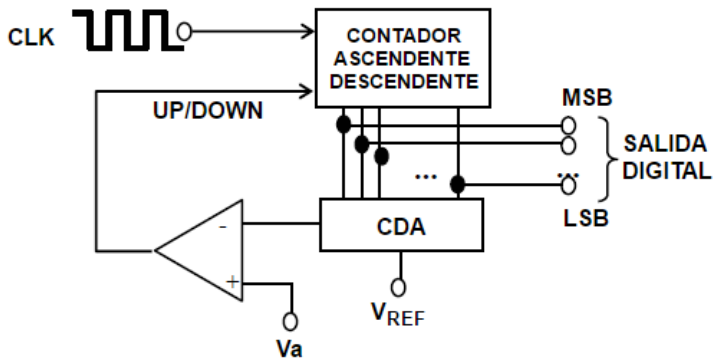


Figura 25: CAD de seguimiento

2.4.4. CAD DE DOBLE RAMPA

En los convertidores de doble rampa se convierte la tensión analógica de entrada en el intervalo temporal que dura la descarga de un condensador, para luego convertir esta magnitud en una salida digital. Dicha conversión se realiza en dos fases. La figura 26 muestra el esquema del circuito. Este circuito es muy lento pero muy preciso; se utiliza generalmente en medidas lentas que requieran precisión, como por ejemplo en los multímetros digitales. Presentan una característica muy lineal, un reducido offset y bajos errores de ganancia.

Veamos el funcionamiento para una entrada analógica unipolar, para Va>0 y -Vref<0.

En la primera fase, se pone el contador en modo decreciente con todas sus salidas a 1 y el integrador se pone a cero (cortocircuitando el condensador mediante un circuito adicional), y se conecta el interruptor S a la tensión analógica que se va a convertir, Va. La salida de la puerta NOR es 0 y Q=1. La salida del integrador es una rampa de ecuación:

$$V_O = -\frac{1}{R_1 \cdot C_1}\int_0^t (V_a)\cdot d\tau = -\frac{1}{R_1 \cdot C_1}\int_0^{T1} (V_a)\cdot d\tau = -\frac{V_a \cdot T_1}{R_1 \cdot C_1}$$

Esta salida se mantiene hasta que todos los bits del contador hayan caído a cero, como se muestra la figura 27. Como la rampa es decreciente, la tensión diferencial en el AO comparador es positiva, y su salida es un nivel alto, que habilita el paso de la señal de reloj por la puerta AND. En consecuencia, esta rampa decreciente tiene siempre la misma duración, $T_1=2^N T_{CLK}$, para cualquier tensión analógica a convertir.

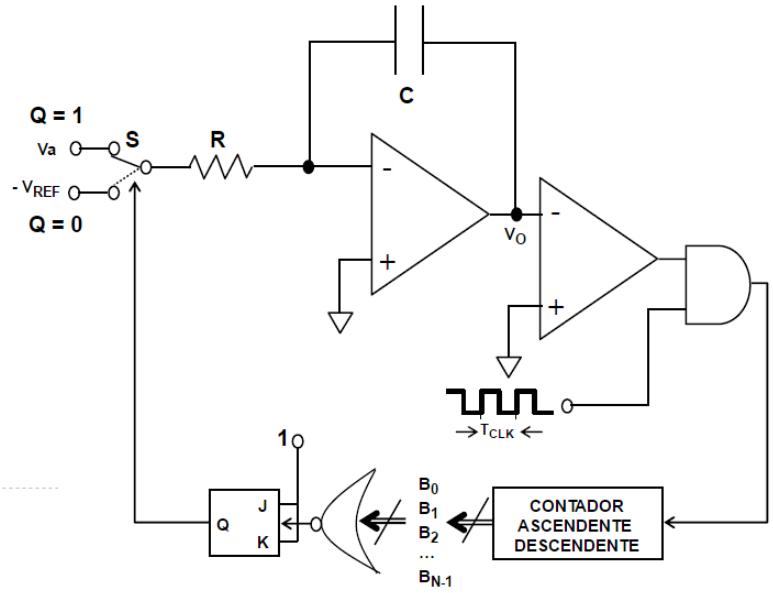


Figura 26: CAD de doble rampa

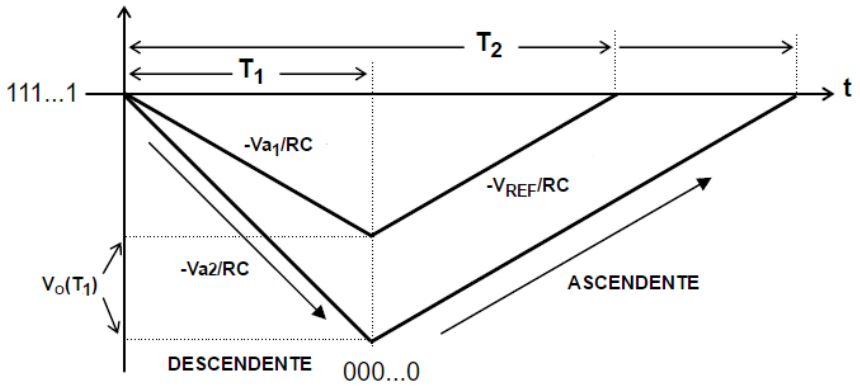


Figura 27: Curva característica de un CAD de doble rampa

En la segunda fase, cuando todas las salidas del contador son nulas (cuando ha finalizado la cuenta decreciente) la salida de la puerta NOR se pone a 1 y Q=0; pasándose a integrar la tensión de referencia, para cualquier tensión a convertir. La ecuación del integrador es en este caso:

$$V_O = V_O(T_1) + \int_{T1}^{t} -V_{ref} \cdot d\tau = -\frac{V_a \cdot T_1}{R_1 \cdot C_1} + \int_{T1}^{T2} -V_{ref} \cdot d\tau =$$

$$= -\frac{V_a \cdot T_1}{R_1 \cdot C_1} + \frac{V_{ref} \cdot (T_2 - T_1)}{R_1 \cdot C_1}$$

Esta rampa creciente termina en el instante T2, cuando la salida del integrador es nula, la tensión diferencial del comparador se anula y su salida pasa a cero, inhibiéndose el reloj. En este instante:

$$0 = -\frac{V_a \cdot T_1}{R_1.C_1} + \frac{V_{ref} \cdot (T_2 - T_1)}{R_1 \cdot C_1} \xrightarrow{T_1=2^N \cdot T_{CLK}} T_2 - T_1 = \frac{V_a}{V_{ref}} \cdot 2^N \cdot T_{CLK}$$

Esto significa que el intervalo de tiempo T_2 - T_1 es proporcional al periodo de reloj. La constante de proporcionalidad es el número de impulsos o cuentas transcurridas hasta que se anula la salida del integrador. Este número decimal permite obtener la palabra digital al codificarlo en binario:

$$T_2 - T_1 = \frac{V_a}{V_{ref}} \cdot 2^N \cdot T_{CLK} \ , con \ \frac{V_a}{V_{ref}} \cdot 2^N = cte$$

Los CADs que integran la señal de entrada pueden rechazar las interferencias que enmascaran la señal de interés. Éstas suelen derivar de la red, por lo que se escoge un múltiplo de dicha frecuencia como periodo de integración con el fin de eliminarlas.

Es muy importante que **la tensión de entrada analógica no supere a la tensión de referencia** ya que puede dar lugar a errores en la señal digitalizada.

Este es el conversor que se va a emplear en el proyecto. En el capítulo siguiente se explicará cual se estructura interna, ya que posee una serie de modificaciones respecto al

circuito explicado, como entrada/salida diferencial, posibilidad de seleccionar Vref/-Va o –Vref/Va, así como el uso de un contador no lineal.

2.4.5. CAD DE APROXIMACIONES SUCESIVAS

Es el más común de los convertidores integrados cuando la exactitud requerida no es determinante, ya que su diseño supone un equilibrio entre velocidad y complejidad. Se caracteriza por incluir un registro de aproximaciones sucesivas (SAR) que contiene las distintas aproximaciones de la palabra digital. La figura 28 muestra el esquema de un CAD de aproximaciones sucesivas de 8 bits. En ella se observa el SAR y la cadena de biestables tipo "D", encargados de propagar un "1" de forma cíclica, desde que D8 recibe el impulso de disparo que inicializa la conversión.

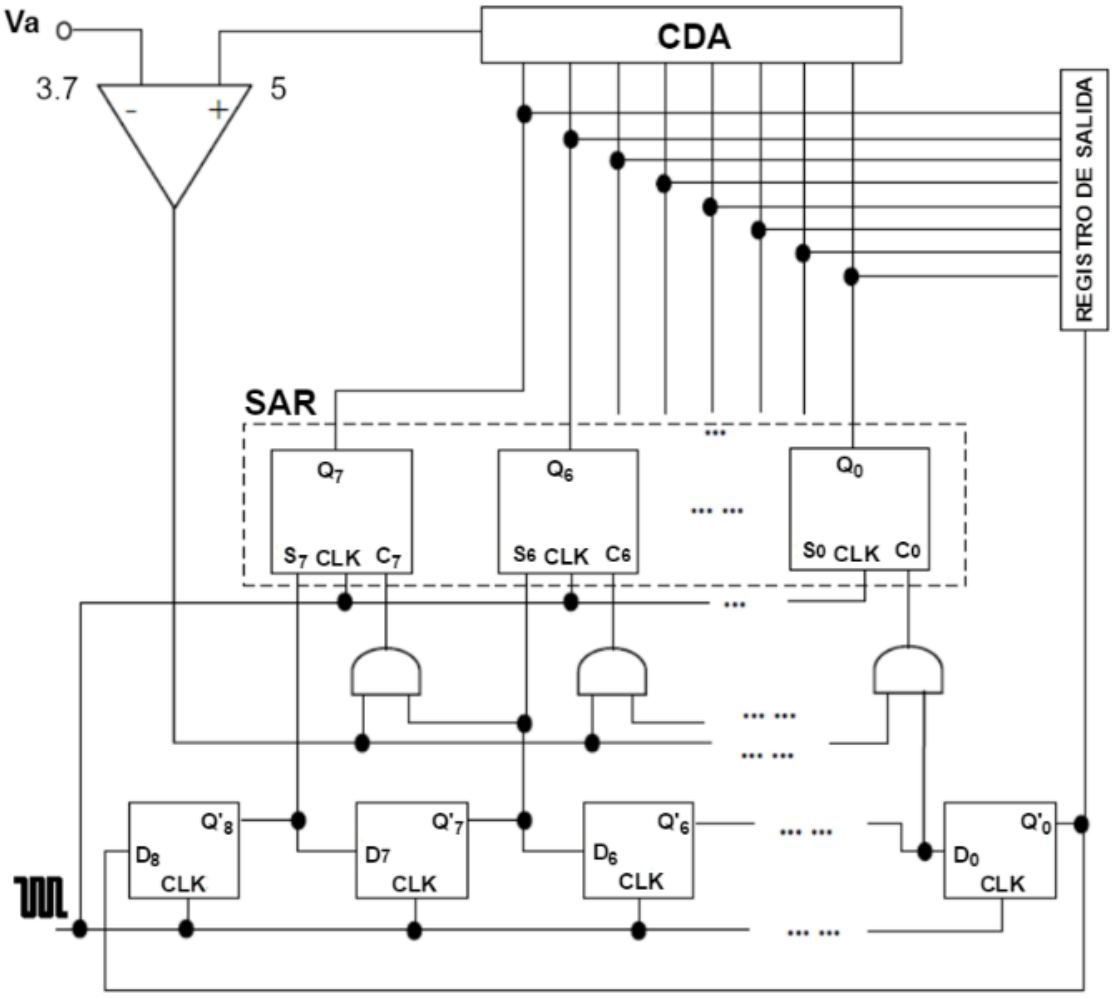


Figura 28: CAD de aproximaciones sucesivas

El funcionamiento se muestra convirtiendo a digital una tensión analógica de 3,7 V sobre un fondo de escala en la entrada de 10 V. Inicialmente se pone a 1 el bit más significativo, Q7=1, manteniendo a cero el resto, y antes de llegar el impulso de disparo a D8, todos los flip-flop "D" ofrecen salida nula. Se convierte a analógica la palabra digital resultante (10000000) y se compara con la señal a convertir (3,7 V). Como la tensión equivalente a la

palabra digital (5 V) es superior, la salida del comparador es un "1"; C7=1 como resultado de la propagación del "1" por la cadena D. Entonces Q7=0 y Q6=1; se convierte a analógica la palabra digital y así sucesivamente hasta que el "1" se ha propagado 8 veces por la cadena D. La tabla 2 muestra el proceso completo de conversión en los 8 ciclos de reloj que transcurren hasta el fin de conversión. Éste se suele anunciar por un terminal dispuesto a tal efecto. El proceso de conversión es propio de un circuito realimentado, en el que se compara la señal a convertir con los distintos acercamientos de la palabra digital.

Pulso	Palabra digital (Qi)	Fracción de estado - Tensión aproximada	Bits del SAR afectados
0, inicio	10000000	(128/256)*10 = 5 > 3.7	Q_7=0 y Q_6=1
1	01000000	(64/256)*10 = 2.5 > 3.7	Q_6=1 y Q_5=1
2	01100000	(96/256)*10 = 3.75 > 3.7	Q_5=0 y Q_4=1
3	01010000	(80/256)*10 = 3.125 > 3.7	Q_4=1 y Q_3=1
4	01011000	(88/256)*10 = 3.43755 > 3.7	Q_3=1 y Q_2=1
5	01011100	(92/256)*10 = 3.593955 > 3.7	Q_2=1 y Q_1=1
6	01011110	(94/256)*10 = 3.671875 > 3.7	Q_1=1 y Q_0=1
7	01011111	(95/256)*10 = 3.7109375 > 3.7	Q_0=0, fin conversión
8	01011110		

Tabla 2: Ejemplo de conversión de una tensión de entrada de 3.7 V para el CAD de aproximaciones sucesivas

Este método de conversión es útil cuando la resolución no es un parámetro que limite en exceso el diseño, ya que ofrece velocidad a bajo coste con resoluciones de 8, 10, 12, 14 y 16 bits. El tiempo de conversión resulta de multiplicar el número de bits más 1 por el periodo del reloj, que suele ser interno al circuito integrado, aunque existen modelos que permiten emplear reloj externo. Esto se debe a que la palabra digital final no pasa al registro de salida hasta el siguiente flanco de reloj, en el que también se informa del fin de la conversión.

2.4.6. CONVERTIDORES SIGMA – DELTA (Σ – Δ)

Son apropiados para aplicaciones con requisitos de resolución elevados (hasta 21 bits en algunos modelos) que involucren frecuencias bajas-medias (audio y voz entre 10 Hz y 100 kHz). El esquema de la siguiente figura muestra la estructura interna de este circuito.

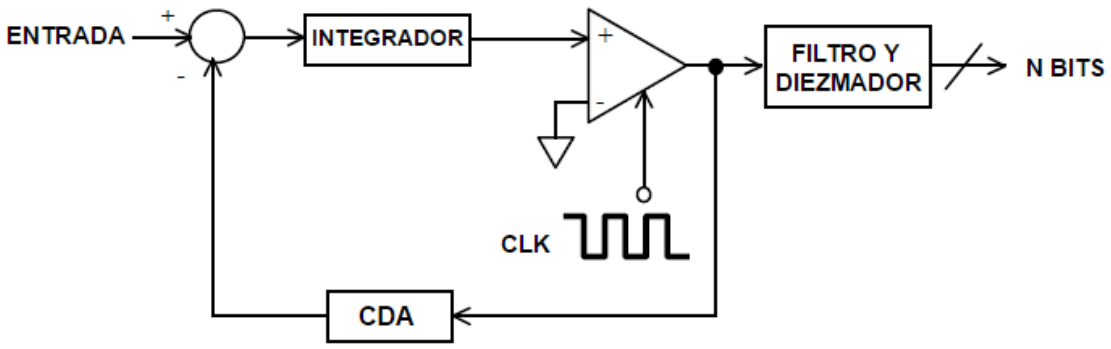


Figura 29: Convertidor Sigma – Delta (Σ – Δ)

El comparador de alta velocidad compara la salida del integrador con cero. El CDA de 1 bit toma el "0" o el "1" de la salida del comparador y genera una tensión analógica que se resta a la señal de interés. La diferencia es integrada y comparada con cero. Por ejemplo, para una entrada positiva, la salida del comparador es una secuencia de "1" hasta que la salida del comparador pasa por cero. Cuanto más positiva sea la entrada mayor es la serie de "1" producida. Para entrada nula, en la salida del comparador se alternan los "1" con los "0".

En este circuito la frecuencia de muestreo puede ser muy elevada comparada con la de la señal de entrada, por lo que el filtro "antialiasing" es muy simple. Tampoco es necesario el circuito S&H.

Como se ha podido comprobar existen diferentes tipos de arquitecturas de conversión, cada una con sus peculiaridades y características propias. Para la realización de la plataforma de implementación y testeo se ha decidido emplear el conversor **integrador de doble rampa**. Este CAD es muy flexible debido a su parte analógica sencilla. Además a través de las señales de habilitación es posible secuenciar fácilmente varios métodos de procesado, lo que lo hace ideal para probar diferentes algoritmos de conversión A/D no lineales. Este CAD es apto para aplicaciones en las que la velocidad no sea un requerimiento crítico, es decir, para señales lentas como las que proporcionan determinados sensores. Además, es un conversor muy preciso dependiendo de la base de tiempo, por lo que lo hace ideal para realizar nuestra plataforma.

En el siguiente capítulo se va a realizar un estudio detallado de cómo está implementado el conversor así como las modificaciones que ha sufrido respecto al anteriormente visto. También se analizará la plataforma Arduino, sus puertos de entrada y salida y sus características más relevantes para el proyecto.

3. CONVERSOR A/D DE DOBLE RAMPA NO LINEAL CONFIGURABLE Y CONTROLADO POR ARDUINO

En el capítulo anterior se ha realizado un estudio completo sobre los CAD. Se ha analizado todo el proceso de conversión de la señal analógica a digital, los errores del conversor así como sus parámetros más relevantes, y por último una comparativa en las diferentes implementaciones de estos. Se ha citado que el tipo de conversor a emplear en la plataforma va a ser el **conversor de doble rampa** debido a su modularidad y facilidad para hacer reconfigurarlo. Como ya se ha citado, este conversor es apto para aplicaciones en las que la velocidad no sea un requerimiento crítico. Además, es un conversor muy preciso, por lo que lo hace ideal para dicha aplicación.

En este capítulo se va a tratar la arquitectura completa de todo el sistema para realizar diferentes algoritmos que no solo permitan la conversión digital, sino que además permita linealizar la entrada. Es lo que se conoce como un **CAD no lineal**. Se describirán los bloques más importantes y como es su interacción entre ellos, dando como resultado una arquitectura flexible, configurable y de propósito general.

3.1. ARQUITECTURA GENERAL

Para tener una perspectiva global del proyecto, se muestra en la siguiente figura el diagrama de bloques de la plataforma.

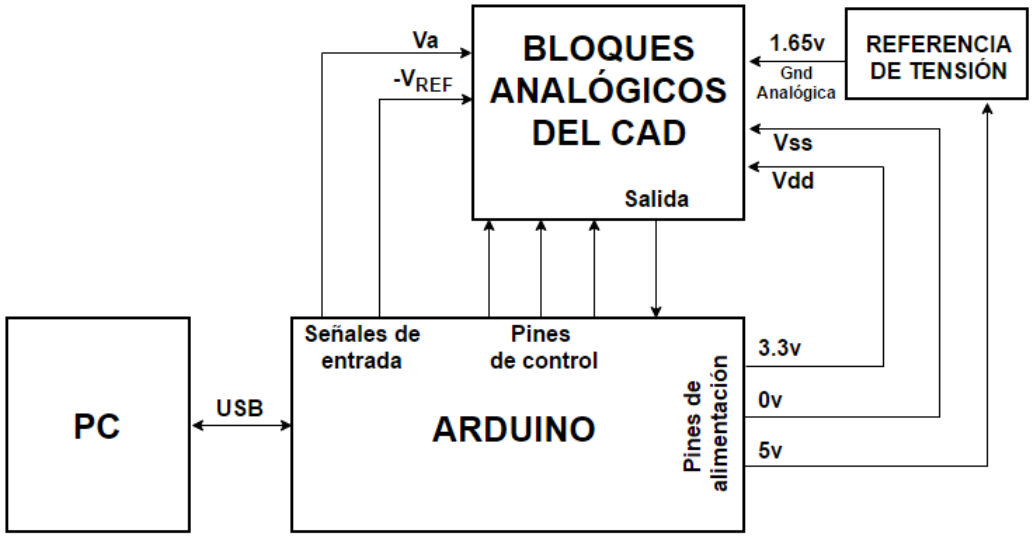


Figura 30: Diagrama de bloques de la plataforma

Como se puede observar el sistema está formado por dos partes, una realizada con un integrado de tecnología de 0.5µm previamente fabricado que contiene la parte analógica del conversor y otra implementada en Arduino. El integrado es el encargado de realizar la integración de la señal y comparar dicha señal integrada con un nivel de referencia, para

indicar que ha finalizado la conversión. También realiza otras funcionalidades como la selección de la señal de entrada, la inversión de la misma o el reseteo del integrador. Por otra parte la arquitectura general se completa usando Arduino, que es el encargado de realizar el control lógico de los diferentes bloques analógicos del chip fabricado, además de realizar los diferentes algoritmos con su CPU

La parte analógica del CAD presenta una serie de pines de control, lo cuales se encargarán de conmutar de la señal analógica de entrada a la señal de referencia, resetear el integrador, invertir la señal de entrada o proporcionar el resultado de la comparación. Los bloques pueden ser habilitados o inhabilitados a través de dichos pines, resultando en una arquitectura flexible y reconfigurable. Dichos pines de control irán conectados a Arduino, el cual se encargará de gestionarlos para realizar las diferentes funciones del convertidor de doble rampa, así como el algoritmo de linealización.

Como cualquier otro sistema, el conversor requiere de una tensión de alimentación para su funcionamiento. Esta se obtendrá de Arduino, el cual posee una serie de puertos que proporcionan unas tensiones estables de alimentación. También se requerirá de una referencia de tensión externa, la cual también irá alimentada por Arduino, y cuya salida irá conectada al integrado. Dicha tensión se empleará como masa del integrado, pero en vez de ser la tradicional masa de 0V, será una masa analógica de 1.65V para el funcionamiento del integrado. El motivo se debe a que se va a alimentar el conversor con una alimentación asimétrica, es decir, con una tensión positiva de 3.3V y una negativa de 0V. Como el integrado requiere de una masa para su funcionamiento, la cual es el valor medio de la tensión positiva y negativa de valor de 1.65V, se ha decidido emplear la referencia de tensión externa. Por último, Arduino va conectado mediante cable USB al ordenador. A través de él se cargan los diferentes algoritmos y se alimenta Arduino.

En los siguientes apartados se va a analizar los dos elementos que forman la plataforma: los **bloques del CAD de doble rampa** y la plataforma **Arduino** para su control. Para el primero, se explicará cómo está implementado el integrado, las modificaciones que se han realizado respecto al esquema habitual así como los diferentes pines que posee. Para el segundo, se estudiará la plataforma Arduino, dando una visión general sobre el sistema y cuáles son las características que servirán para llevar a cabo dicha plataforma.

3.2. ANÁLISIS DEL INTEGRADO

El CAD de doble rampa ha sido diseñado sobre un encapsulado DIP 40. El circuito completo, además del integrador y el comparador, posee una serie de elementos adicionales tales como una red de conmutadores para seleccionar las tensiones de entrada así como unos buffers en configuración inversora, o el empleo de señales diferenciales en el procesado. Las diferencias que presenta el circuito con respecto al de la figura 26 son las siguientes:

- Tensión de entrada y salida diferenciales (fully – diferential)
- Posibilidad de seleccionar +Va/-Vref y –Va/+Vref

También posee una fuente de corriente controlada de manera externa que polariza las diferentes etapas del integrado. En la siguiente figura se observa como es la estructura interna del sistema. Los números representan los diferentes pines del circuito, y los diferentes componentes forman la estructura interna del conversor excepto las resistencias y condensadores que son externos a este, para poder configurar valores como ganancia, corrientes de polarización y pendiente de integración. Los puntos en rojo indican los pines que van a ser conectados a los puertos de Arduino.

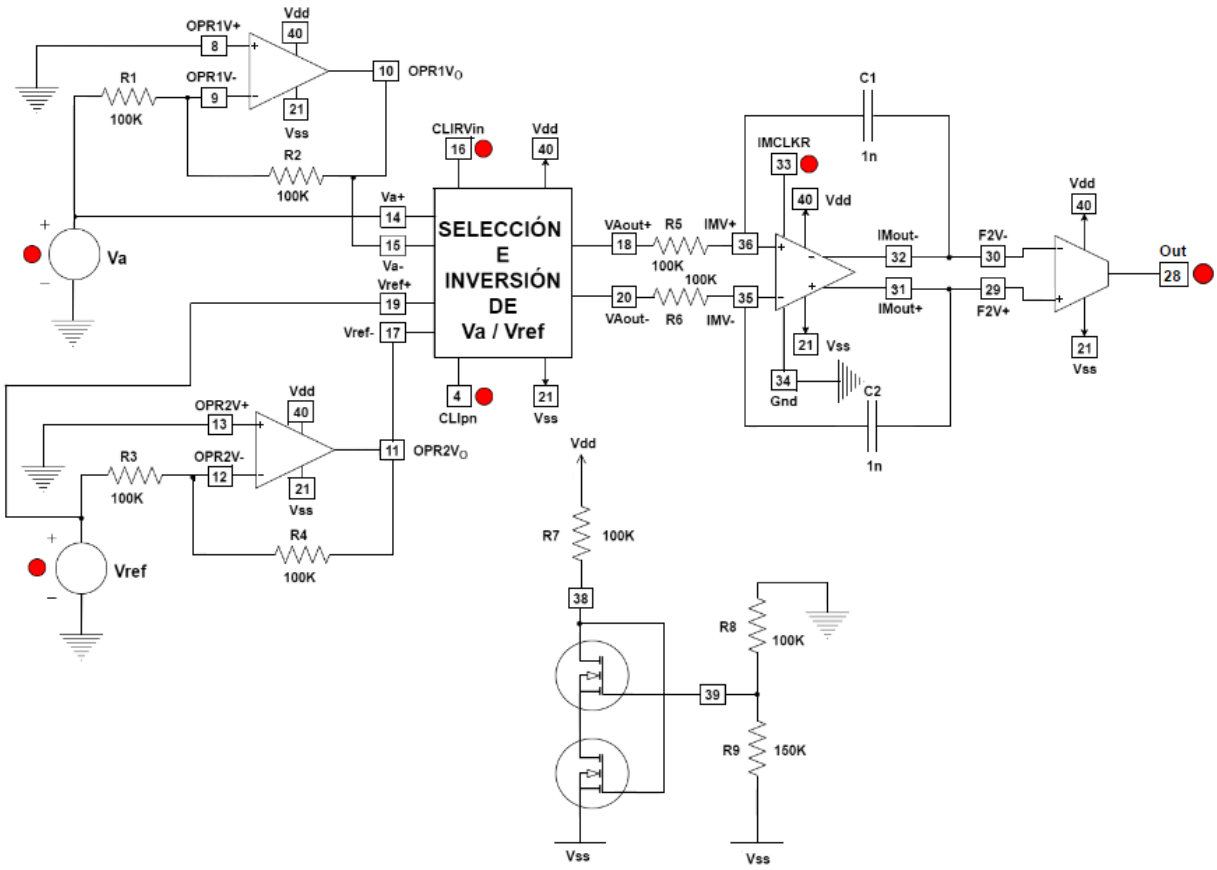


Figura 31: Arquitectura interna del integrado junto con sus pines de conexión

La tensión Va corresponde a la tensión analógica de entrada, procedente de un sensor o de cualquier otro elemento generador de señal. Esta tensión se introduce en el pin 14 (**Vin+**). También se introduce en un amplificador en configuración no inversora con ganancia unidad atendiendo a la relación de resistencias. De esta manera, obtenemos a su salida la tensión de entrada analógica invertida –Va, la cual está conectada al pin 15 (**Vin–**). Este mismo procedimiento es aplicado a Vref, el cual se corresponde con la tensión de referencia. Se introduce en el pin 19 (**Vref+**), así como a un amplificador en configuración no inversora, como en el caso anterior, obteniendo a su salida la tensión de referencia invertida –Vref, que a su vez se conecta al pin 17 (**Vref–**). De esta manera, obtenemos las tensiones necesarias para poder efectuar el proceso de conversión, el cual puede observarse en la figura 27. La ventaja de este circuito es que permite establecer dos gráficas de

conversión, dependiendo de cuál sea las polaridades de entrada. Podrán establecerse por lo tanto dos casos:

- Tensión analógica de entrada +Va y tensión de referencia –Vref
- Tensión analógica de entrada –Va y tensión de referencia +Vref

En la gráfica de la figura 27 se ha tomado el primer caso, obteniendo una señal con forma de triángulo invertido y tomando primero un contador decreciente y posteriormente creciente. Si seleccionamos el segundo caso, obtenemos la misma señal sólo que de carácter positivo. El primer contador será creciente y el segundo decreciente. La filosofía de comparación es la misma para ambos casos, siendo el instante de cruce por cero el que marca el fin de la conversión.

Para poder seleccionar uno de los dos posibles casos de conversión, se ha implementado una red de conmutadores. Dicha red de conmutadores puede observarse en la siguiente figura.

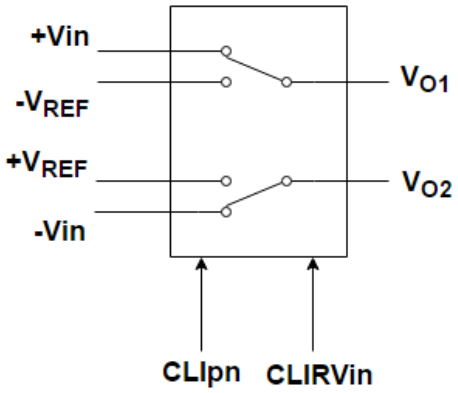


Figura 32: Red de conmutadores del integrado junto con sus pines de selección e inversión de señal

Como se observa, dicha red selecciona la tensión Va o Vref, obteniendo a la salida una señal diferencial. Mediante el pin 16 (**CLIRVin**) seleccionamos una u otra. Podemos invertir dichas señales a la salida de la red de conmutadores mediante el pin 4 (**CLIpn**). En la siguiente tabla se resume el funcionamiento de dicha red.

CLIpn	CLIRVin	Vo1	Vo2
0	0	–Vref	+Vref
0	1	+Vref	–Vref
1	0	–Va	+Va
1	1	+Va	–Va

Tabla 3: Tabla de verdad del funcionamiento de la red de conmutadores

A partir de aquí tenemos una tensión diferencial, la cual se introduce en el integrador. Como hemos citado, la modificación de este conversor con respecto al de la figura 26 es que trabaja con señales diferenciales. Por lo tanto, la tensión de entrada del integrador así

como su salida también será diferencial. El proceso de integración es idéntico al explicado en el apartado 2.4.4., siendo las ecuaciones las mismas. No obstante, se van a incluir a continuación para establecer una comparación entre los dos casos.

- Tensión analógica de entrada +Va y tensión de referencia –Vref:

$$V_O = -\frac{1}{R_1 \cdot C_1} \cdot \int_0^t (V_a) \cdot d\tau = -\frac{1}{R_1 \cdot C_1} \cdot \int_0^{T1} (V_a) \cdot d\tau = -\frac{V_a \cdot T_1}{R_1 \cdot C_1}$$

$$V_O = V_O(T_1) + \int_{T1}^t -V_{ref} \cdot d\tau = -\frac{V_a \cdot T_1}{R_1 \cdot C_1} + -\frac{1}{R_1 \cdot C_1} \int_{T1}^{T2} -V_{ref} \cdot d\tau =$$

$$= -\frac{V_a \cdot T_1}{R_1 \cdot C_1} + \frac{V_{ref} \cdot (T_2 - T_1)}{R_1 \cdot C_1}$$

- Tensión analógica de entrada –Va y tensión de referencia +Vref

$$V_O = -\frac{1}{R_1 \cdot C_1} \int_0^t (-V_a) \cdot d\tau = \frac{1}{R_1 \cdot C_1} \int_0^{T1} (V_a) \cdot d\tau = \frac{V_a \cdot T_1}{R_1 \cdot C_1}$$

$$V_O = V_O(T_1) + \int_{T1}^t V_{ref} \cdot d\tau = -\frac{V_a \cdot T_1}{R_1 \cdot C_1} + -\frac{1}{R_1 \cdot C_1} \cdot \int_{T1}^{T2} V_{ref} \cdot d\tau =$$

$$= \frac{V_a \cdot T_1}{R_1 \cdot C_1} - \frac{V_{ref} \cdot (T_2 - T_1)}{R_1 \cdot C_1}$$

Cabe remarcar que mediante las resistencias externas R5 y R6 así como los condensadores C1 y C2 se controla la pendiente de integración.

Una vez integrada la señal y obteniendo una salida diferencial, es comparada con una tensión de referencia. De esta manera, obtenemos el instante en el que se produce el cruce por cero y en consecuencia el fin de la conversión. La idea de comparación es la misma que la del apartado 2.4.4., sólo que la entrada del comparador es diferencial.

Otro detalle a destacar es el pin 33 (**IMCLKR**), el cual realiza el reseteo del integrador cortocircuitando el condensador y con ello poniendo su salida a cero.

Por último, el pin 21 es el de alimentación negativa (**Vss**) y el pin 40 es el de alimentación positiva (**Vdd**). Para poder polarizar las fuentes de corriente que forman los amplificadores del circuito, se emplea una fuente de corriente, la cual se polariza con la resistencia R7 alimentada por Vdd y conectada al pin 38 (**IinBB**) así como las resistencias R8 y R9 conectadas de manera externa al pin 39 (**VVBB**).

Con respecto a la tensión de alimentación del integrado, se ha visto que se alimenta con una tensión positiva y negativa, es decir, con una tensión simétrica. El sistema está diseñado para operar a una tensión ± 3.6V, pero puede funcionar a tensiones superiores e inferiores. De manera experimental, se han hecho pruebas desde ± 1.8V hasta ± 7V. En esos casos, los pines 8, 13 y 34 están conectados a masa (0V). También se puede alimentar

el integrado de manera asimétrica, es decir, únicamente con tensiones positivas. En ese caso, el pin Vdd estará conectado a la tensión positiva, Vss a una tensión de 0V, y la masa del integrado (pines 8, 13 y 34) se conectará a una "**tierra analógica**", cuyo valor será la mitad de la tensión de alimentación. De cara a diseñar el PCB, se tomará esta configuración ya que Arduino proporciona únicamente tensiones positivas. Este tema se tratará con más detalle en el capítulo siguiente, pero se ha hecho una pequeña introducción con el fin de explicar todos los detalles del integrado.

En la siguiente figura se muestran los pines del integrado. En la tabla se explica la funcionalidad de cada pin.

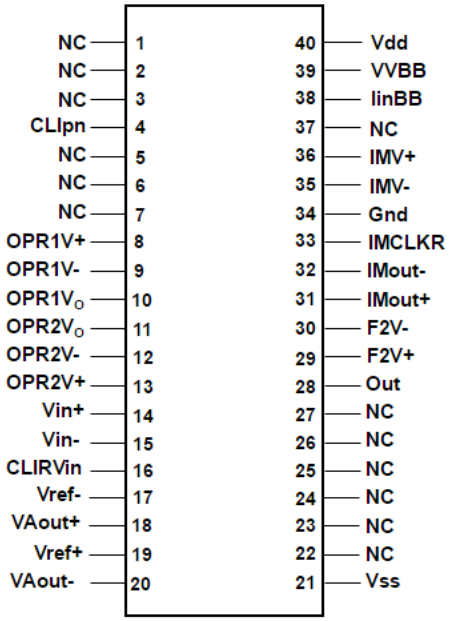


Figura 33: Encapsulado del integrado junto con sus pines de conexión

PIN		E/S	DESCRIPCIÓN
NOMBRE	Nº		
NC	1		No conectado, debe dejarse flotando
NC	2		No conectado, debe dejarse flotando
NC	3		No conectado, debe dejarse flotando
CLIpn	4	E	Inversión de las señales de entrada; Inversión("1") y no inversión ("0")
NC	5		No conectado, debe dejarse flotando
NC	6		No conectado, debe dejarse flotando
NC	7		No conectado, debe dejarse flotando
OPR1V+	8	E	Entrada no inversora del amplificador 1
OPR1V-	9	E	Entrada inversora del amplificador 1
OPR1Vo	10	S	Salida del amplificador 1
OPR2Vo	11	S	Salida del amplificador 2
OPR2V-	12	E	Entrada inversora del amplificador 2
OPR2V+	13	E	Entrada no inversora del amplificador 2
Vin+	14	E	Tensión de entrada Va

Vin-	15	E	Tensión de entrada -Va
CLIRVin	16	E	Selección de Va ("1") o Vref ("0")
Vref-	17	E	Tensión de entrada Vref
VAout+	18	S	Tensión diferencial de salida de la red de conmutadores
Vref+	19	E	Tensión de entrada -Vref
VAout-	20	S	Tensión diferencial de salida de la red de conmutadores
Vss	21	E	Tensión de alimentación negativa
NC	22		No conectado, debe dejarse flotando
NC	23		No conectado, debe dejarse flotando
NC	24		No conectado, debe dejarse flotando
NC	25		No conectado, debe dejarse flotando
NC	26		No conectado, debe dejarse flotando
NC	27		No conectado, debe dejarse flotando
Out	28	S	Salida del comparador
F2V+	29	E	Tensión diferencial de entrada al comparador
F2V-	30	E	Tensión diferencial de entrada al comparador
IMout+	31	S	Tensión diferencial de salida del integrador
IMout-	32	S	Tensión diferencial de salida del integrador
IMCLKR	33	E	Reseteo del contador; reseteo ("1"), no reseteo("0")
Gnd	34	E	Masa del integrado
IMV-	35	E	Tensión diferencial de entrada al integrador
IMV+	36	E	Tensión diferencial de entrada al integrador
NC	37		No conectado, debe dejarse flotando
IinBB	38	E	Corriente de polarización
VVBB	39	E	Corriente de polarización
Vdd	40	E	Tensión de alimentación positiva

Tabla 4: Resumen de los pines del integrado junto con su funcionalidad

3.3. ANÁLISIS DE ARDUINO DUE

En la introducción del proyecto, se ha comentado que se va a usar Arduino como elemento de control del integrado, el cual posee únicamente la parte analógica del conversor. En el aparatado de análisis del integrado, se ha visto que hay una serie de pines que controlan el CAD, tales como el pin 4 CLIpn (Inversión de la señal), el pin 16 CLIRVin (Selección de Va o Vref) y el pin 33 IMCLKR (Reseteo del integrador). La gestión de estos pines la va llevar a cabo Arduino proporcionando flexibilidad y posibilidad de configurar diversos algoritmos de conversión.

Arduino es una plataforma de hardware libre basada en una placa con un microcontrolador, la cual proporciona toda la electrónica necesaria para ser configurada mediante un entorno de desarrollo. El modelo de Arduino que se va a emplear es el "Arduino Due", el cual es uno de los más potentes del mercado en cuanto a características se refiere.

Arduino Due es una placa basada en el microcontrolador de Atmel® **SAM3X8E ARM Cortex-M3 CPU**. Es la primera placa que está basada en un microcontrolador de 32 bits ARM[1]. Posee 54 pines de entrada/salida digitales (de las cuales 12 pueden emplearse con PWM, *Pulse Width Modulation* o Modulación de la Anchura de Pulsos), 12 entradas analógicas, 4 UARTs, un reloj de 84MHz así como dos CDA. También presenta otras características como un botón de reseteo del sistema, conectores I^2C o un pin de alimentación.

La placa posee toda la electrónica necesaria para funcionar. Conectándola al ordenador mediante un cable USB, nos comunicamos con ella para subirle los diferentes Sketches[2] y le proporcionamos la tensión de alimentación. También podemos alimentarlo mediante un conversor AC/DC, conectándolo al pin de alimentación. En la siguiente figura se observa la placa Arduino Due.

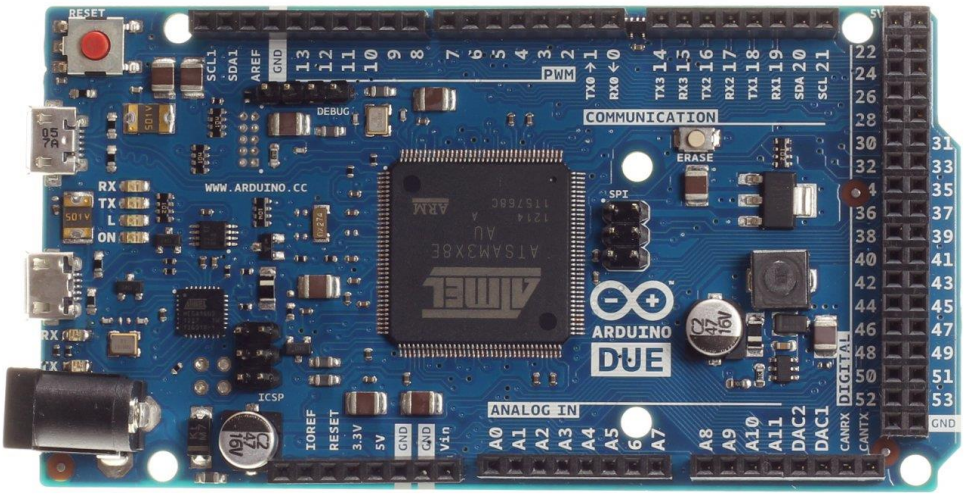


Figura 34: Arduino Due

Una cuestión a tener en cuenta es que la placa funciona a 3.3V. El voltaje máximo que soportan las entradas y salidas es 3.3V. Aplicar tensiones por encima de él puede destruir la entrada o salida.

A continuación se van a explicar los aspectos más importantes de la placa, para que sirvan de referencia a la hora de hacer el diseño.

1 La arquitectura **ARM** es el conjunto de instrucciones de 32 y 64 bits basado en una arquitectura RISC desarrollada por ARM Holdings. Un diseño basado en RISC implica que los procesadores ARM necesitan una cantidad menor de transistores que los procreadores x86 CISC típicos en la mayoría de ordenadores personales. La relativa simplicidad de los procesadores ARM los hace ideales para aplicaciones de baja potencia.

2 Un **Sketch** es el nombre que emplea Arduino para un programa. Es la unidad que se descarga y ejecuta en la placa.

ARM CORE

Un núcleo de 32 bits, que permite realizar operaciones de 4 bytes. Esto implicará poder realizar cálculos de mayor precisión que el resto de Arduinos, en los cuales los microcontroladores son de 1 byte. La frecuencia de operación es de 84MHz. Posee además una memoria flash de 512KBytes (dos bloques de 256KBytes) para almacenar código, así como 96KBytes (un bloque de 64KBytes y otro de 32KBytes) de memoria SRAM.

ALIMENTACIÓN

Arduino Due puede ser alimentado por el conector USB o mediante una fuente de alimentación externa. En el caso del USB, la tensión que maneja es de 5V. La placa posee un regulador que convertirá dicha tensión a 3.3V, ya que es la tensión a la que opera el microcontrolador. En el caso de usar la fuente externa, el rango de voltaje recomendado está entre 7 y 12V. El regulador convertirá la tensión a 3.3V.

Los pines de alimentación son los siguientes:

- **Vin**: Es la entrada a la cual le conectamos la fuente de alimentación externa. Se puede conectar la fuente a través de dicho pin o a través del Jack de alimentación, el cual está conectado a dicho pin.
- **5V**: Es un pin de salida que proporciona una tensión regulada de 5V. La corriente máxima que puede proporcionar es de 800mA.
- **3.3V**: Es un pin de salida que proporciona una tensión regulada de 3.3V. La corriente máxima que puede proporcionar es de 800mA.
- **GND**: Masa del circuito.

ENTRADAS Y SALIDAS

- **Entradas – salidas digitales**: pines 0 a 53. Cada uno de los pines puede ser usado como entrada o salida dependiendo de cómo los configuremos. Operan a 3.3V.
- **Comunicaciones serie**: Serial 0: (RX) y 1 (TX), Serial 1: 19 (RX) y 18 (TX), Serial 2: 17 (RX) y 16 (TX), Serial 3: 15 (RX) y 14 (TX). Se emplean para recibir o transmitir datos TTL a través del puerto serie, con niveles de 3.3V.
- **Salidas PWM**: pines 2 a 13. Proporcionan una salida PWM de 8 bits.
- **Entradas analógicas**: pines A0 a A11. Permiten introducir señales analógicas al microcontrolador, el cual se encargará de digitalizarlas. Puede proporcionar una resolución de 12 bits, siendo por defecto la de 10 bits.
- **Salidas analógicas**: pines DAC0 y DAC1. Son las salidas de dos conversores analógico – digitales, proporcionándonos una tensión analógica con una resolución entre 8 y 12 bits, la cual es configurable.
- **Referencia de tensión**: pin AREF. Es el voltaje de referencia que se emplea para las entradas analógicas.
- **Pulsador de reseteo**: Resetea el microcontrolador, volviendo a ejecutar el código desde el principio.

PROGRAMACIÓN

Arduino Due se programa mediante un entorno de desarrollo (IDE). En él se realiza la programación de los skectches, así como su compilación y descarga a la placa. Dicho código se almacena en la memoria flash, la cual debe ser borrada cada vez que reprogramamos el microcontrolador. Dicho proceso se realiza de manera automática cada vez que cargamos un sketch a la placa.

Arduino Due posee dos puertos serie para su programación:

- **Puerto de programación** (*Programming Port*): Este puerto es usado cuando queremos programar Arduino de manera normal, es decir, descargarle un sketch para que este haga una funcionalidad determinada.
- **Puerto nativo** (*Native Port*): Usaremos este puerto cuando le conectemos a Arduino Due un dispositivo externo, como un teclado o un ratón y lo controlemos a través de él.

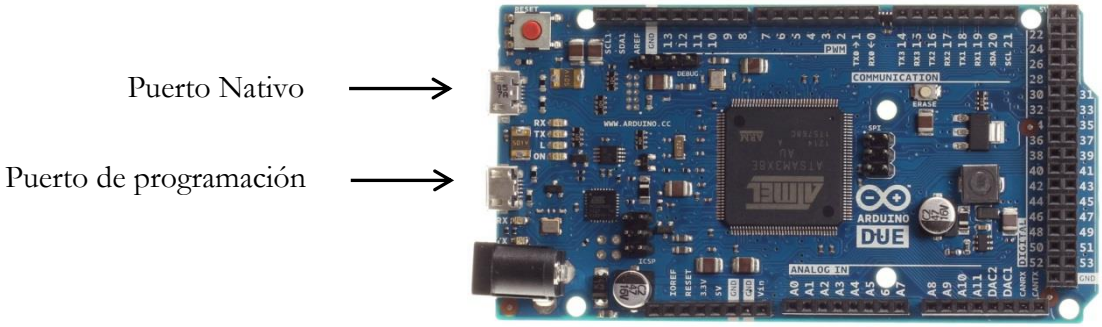


Figura 35: Puerto de programación y nativo de Arduino Due

En la siguiente tabla se resumen las características técnicas del sistema.

Microcontrolador	AT91SAM3X8E
Voltaje de operación	3.3V
Pines digitales de E/S	54 (de los cuales 12 pueden usarse como salida PWM)
Pines analógicos de entrada	12
Pines analógicos de salida	2
Corriente total en todas los pines de E/S	130 mA
Corriente DC para el pin de 3.3V	800 mA
Corriente DC para el pin de 5V	800 mA
Frecuencia del reloj	84 MHz
Memoria flash	512KB
SRAM	96 KB (dos bloques: 64KB y 32KB)

Tabla 5: Resumen de las características de Arduino Due

3.3.1. ENTORNO DE PROGRAMACIÓN

Como hemos citado anteriormente, la programación de Arduino Due y de cualquier Arduino en general, se realiza mediante un entorno de desarrollo IDE. En el escribimos el sketch para que realice una función determinada, realizamos la compilación del mismo así como la descarga al microcontrolador.

En la siguiente figura se muestra una imagen del entorno de programación el cual es enteramente open software y gratis de utilizar. Además se da una descripción breve de sus diferentes funcionalidades.

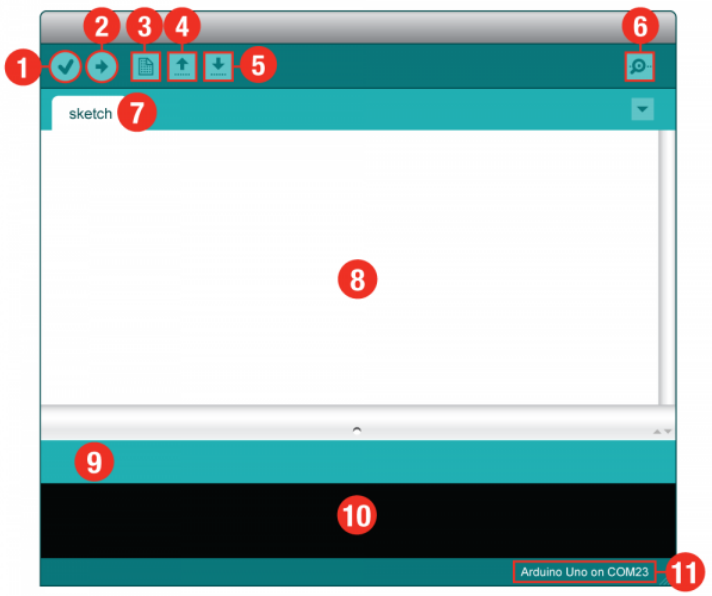


Figura 36: Entorno de programación de Arduino

1. **Verificar**. Compila el código y comprueba si hay errores en el mismo.
2. **Subir**: Además de compilar el código, realiza la descarga del mismo a la placa.
3. **Nuevo**: Crea un nuevo sketch.
4. **Abrir**: Abre un sketch.
5. **Salvar**: Guarda el sketch actual.
6. **Monitor Serie**: Abre una ventana que muestra la información serie que está transmitiendo el microcontrolador. Es muy útil para depurar el código.
7. **Nombre del Sketch**: Nombre del sketch con el que se está trabajando.
8. **Área de código**: Área en la cual se realiza la programación del sketch.
9. **Área de mensajes**: Muestra si hay algún error en el código.
10. **Consola de texto**: Muestra los errores producidos en la compilación. Es muy útil para depurar el código.
11. **Placa y monitor serie**: Indica el modelo de placa que se está usando, así como el puerto serie que está empleando.

Una vez que hemos abierto el entorno de desarrollo, previamente a empezar a escribir código, se deben configurar dos parámetros:

- Selección del **puerto serie**.
- Selección del **modelo** de Arduino.

Es muy importante configurar dichos parámetros ya que si no, no podremos comunicarnos con el microcontrolador.

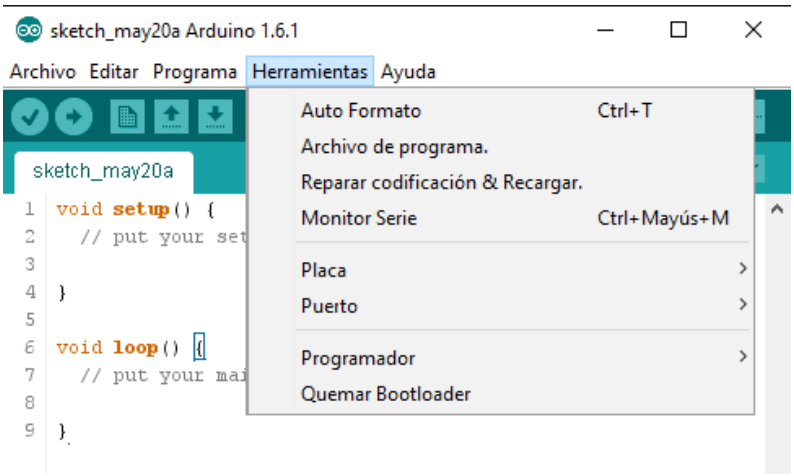


Figura 37: Selección del tipo de placa así como del puerto de programación

A partir de aquí, ya se puede empezar a escribir el código de nuestra aplicación. Como se puede observar en la imagen anterior, en el área de código aparecen dos funciones: el **set up( )** y el **loop( )**. En la primera se deben escribir todos los parámetros de configuración del microcontrolador, tales como la definición de las entradas y salidas, configuración de la resolución de las entradas analógicas y salidas digitales, registros, etc. por ese motivo se llama "set up". Esta función sólo se ejecuta una vez. En la segunda, escribimos el sketch en sí. Se llama "loop" porque se ejecuta de manera indefinida.

Cuando se ejecuta un sketch, primero ejecuta el set up, en el cual configurará los parámetros que le indiquemos. Una vez que haya terminado su ejecución, entrará en el loop y la ejecutará de manera indefinida. Irá ejecutando las líneas que se hayan escrito. Una vez que haya ejecutados todas, volverá a ejecutarlas de nuevo y así sucesivamente.

3.3.2. TIMERS

En este proyecto final de grado se va a utilizar a Arduino para que realice la conversión analógica digital de una arquitectura de doble rampa, que a su vez depende de un contador con una base de tiempo estable y configurable; es por esto que el uso de **TIMERS** se hace imprescindible para realizar las diferentes conversiones. En esta sección se hará una revisión adecuada sobre ellos.

En un microcontrolador, un **timer** es una función que se ejecuta tras un número determinado de ciclos de reloj. Es muy importante hacer una mención a él así como

explicar su funcionamiento, ya que para realizar el control de los pines del integrado se va a hacer uso de él.

A diferencia de los microcontroladores habituales, los cuales suelen poseer tres timers (Timer 0, 1 y 2), el microcontrolador de Arduino emplea una filosofía diferente. Presenta **nueve timers** de propósito general. Están organizados en tres bloques (**TC0, TC1 y TC2**), y cada uno posee a su vez tres canales (**0, 1 y 2**). Cada bloque y cada canal poseen entradas de reloj así como entradas y salidas que pueden emplearse con diferentes propósitos, tales como PWM, contadores, etc.

La ventaja que posee el microcontrolador es que cada timer puede ser programado de manera independiente. Cada canal contiene tres entradas externas de reloj, cinco entradas internas de reloj y dos entradas – salidas de propósito general actuando como un disparo de evento. En la siguiente figura se muestra el diagrama de bloques del timer.

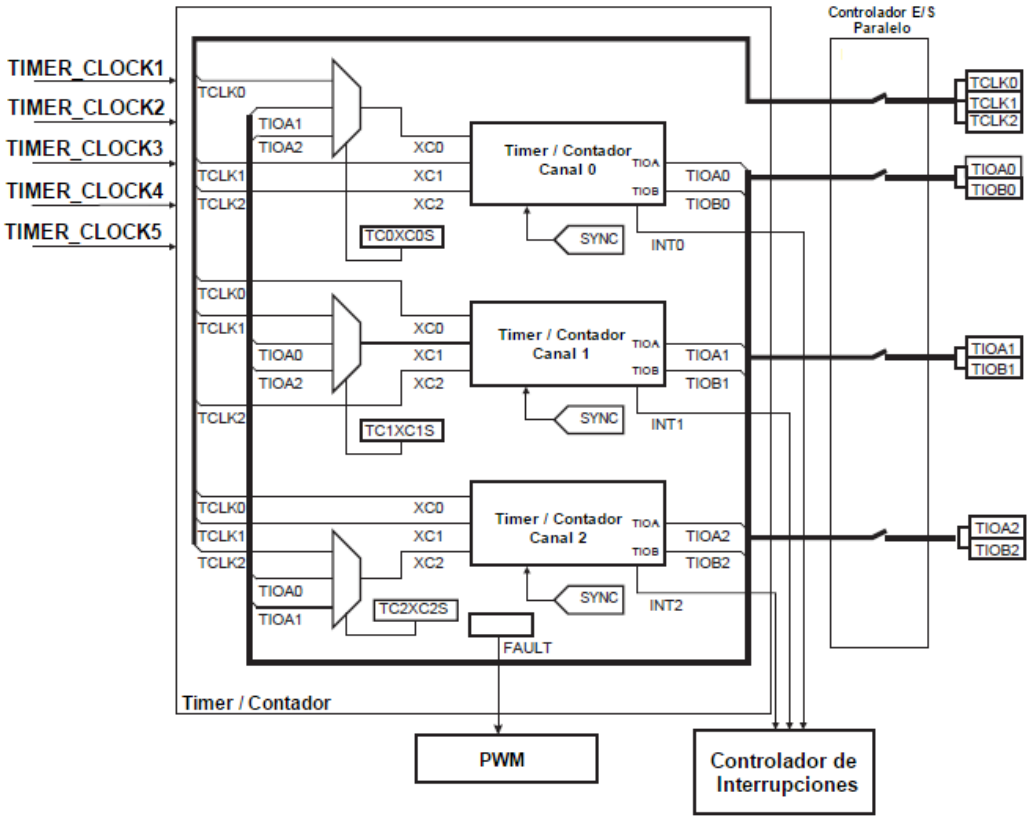


Figura 38: Diagrama de bloques del Timer

Las entradas externas de reloj (**TCLK 0, 1 y 2**) están conectados a unos pines determinados del microcontrolador, en los que introduciremos la señal de reloj de manera externa. Las entradas internas de reloj están conectadas al reloj principal del microcontrolador (*MCK, Master Clock*, a 84MHz). Mediante un divisor de frecuencia se consigue obtener las cinco señales de reloj (**TIMER_CLOCK 1, 2, 3, 4 y 5**), dividiendo la señal de 84MHz por una serie de factores (2, 8, 32, 128 y SCLK). Se podrá seleccionar una u otra por configuración en el set up del sketch. En la figura 39 se muestra el esquema para la selección del reloj. Las entradas y salidas **TIOA** y **TIOB** se emplean para llevar señales procedentes de

dispositivos externos. Cada una de ellas soporta dos modos de funcionamiento: el modo captura (Capture Mode) y el modo forma de onda (Waveform Mode). El primero se emplea para actuar sobre medidas tales como conteo de pulsos, frecuencia, fase, etc. El segundo se emplea para generar señales, como PWM.

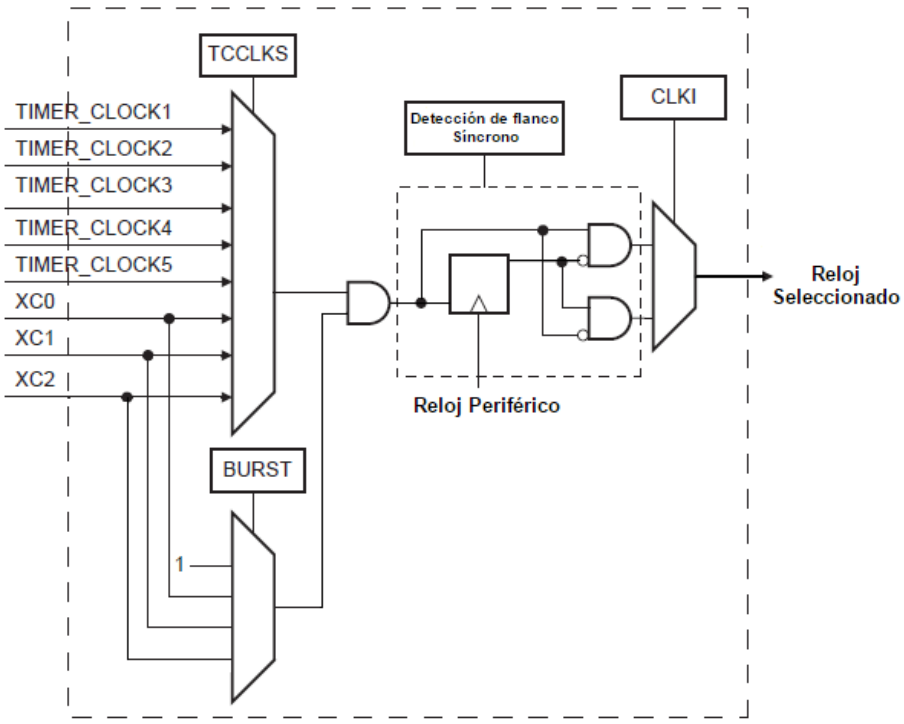


Figura 39: Selección de reloj del Timer

En las siguientes tablas podemos consultar los pines que configuran las entradas externas de reloj así como los valores de las frecuencias divididas del reloj principal.

Entrada de reloj	Puerto del µC	Pin en Arduino Due
TCLK0	PB 26	Pin Digital 22
TCLK1	PA 4	Entrada Analógica 5
TCLK2	PA 7	Pin Digital 31
TCLK3	PA 22	Entrada Analógica 3
TCLK4	PA 23	Entrada Analógica 2
TCLK5	PB 16	DAC1
TCLK6	PC 27	/
TCLK7	PC 30	LED "RX"
TCLK8	PD 9	Pin Digital 30

Tabla 6: Pines de configuración de las entradas externas

Nombre	Definición
TIMER_CLOCK1	MCK / 2
TIMER_CLOCK2	MCK / 8
TIMER_CLOCK3	MCK / 32
TIMER_CLOCK4	MCK / 128
TIMER_CLOCK5	SLCK

Tabla 7: Valores de frecuencia del Timer

NOTA: MCK = 84MHz; SLCK es la frecuencia más baja que puede generar (32768 KHz).

Cada canal emplea un **contador de 32 bits**. El valor de dicho contador se incrementa con cada flanco positivo del reloj seleccionado. Cuando el contador llega a su valor máximo (2^{32}- 1), se resetea y se indica mediante un "flag" que se ha desbordado el contador. Otra forma de resetearlo es mediante un disparo configurable por el usuario. Este contador es útil para realizar conteos de precisión.

En este apartado se ha pretendido explicar el funcionamiento del timer, así como sus modos de funcionamiento, sin entrar en el detalle del código en sí. En el apartado 4.1.4 se explicará más detalladamente como se realiza la configuración de los mismos.

3.3.3. CONTROL DEL CONVERSOR A/D MEDIANTE ARDUINO DUE

Para terminar con el estudio de la plataforma Arduino, se va a hacer una mención a la forma en la que Arduino va a controlar al conversor. El conversor presenta una serie de pines de entrada y salida los cuales van a ser controlados por Arduino.

- **Pin 4 (CLIpn)**: Encargado de invertir la señal de entrada seleccionada. Dado que posee dos estados posibles (inversión y no inversión), se va a controlar mediante un pin digital de Arduino, el cual se configurará como salida.
- **Pin 16 (CLIRVin)**: Encargado de seleccionar la tensión analógica Va o la tensión de referencia Vref. Al igual que el caso anterior, posee dos estados posibles por lo que va a ser controlado por un pin digital de Arduino en configuración de salida.
- **Pin 33 (IMCLKR)**: Encargado de resetear el integrador. Posee dos estados (reseteo y no reseteo) por lo que va a ser controlado por un pin de Arduino en configuración de salida.
- **Pin 28 (OUT)**: Salida del comparador. La salida estará a nivel alto o bajo dependiendo de la comparación, la cual marcará el cruce de paso por cero y con ello el fin de la conversión. Dicha finalización se dará cuando de un valor bajo, por lo que es importante medir dicho nivel. Para ello se empleará un pin digital de Arduino en configuración de entrada, con el fin de leer el valor que nos proporciona el comparador.

- **Pin 21 (Vss)**: Alimentación negativa del conversor. Anteriormente se ha citado que estará conectado a un valor de 0V, por lo que lo conectaremos al pin GND de Arduino, el cual está a dicha tensión.
- **Pin 40 (Vdd)**: Alimentación positiva del conversor. Estará conectado a una tensión de 3.3V, la cual nos la proporciona el pin 3.3V de Arduino.
- **Pines 8, 13 y 34**: Estos pines están conectados a la "tierra analógica" anteriormente mencionada. Poseerá un valor mitad entre 0V y 3.3V, es decir, de 1.65V. Para obtener dicha tensión emplearemos una referencia de tensión.
- **Pin 14 (Vin+) y pin 19 (Vref+):** Pines de entrada del conversor con la tensión +Va y +Vref. Dichas tensiones la vamos obtener con las salidas analógicas de Arduino, es decir con el DAC0 y con el DAC1. Cabe remarcar que los pines 15 (Vin–) y 17 (Vref–) provienen de la salida de los amplificadores inversores, por lo que no estarán conectados a ningún pin de Arduino.

Como conclusión final del presente capítulo, decir que se han analizado los dos elementos que forman la plataforma: el integrado con la parte analógica del conversor y el Arduino Due encargado del control de este. Se han analizado cada uno de ellos por separado, haciendo una referencia a la integración de ambos, indicando los pines que van a ser controlados. En el siguiente capítulo se va a explicar todo el proceso de diseño de la plataforma, desde la elaboración del PCB o shield, hasta la implementación de los algoritmos. También se explicará con más detalle que pines se van a emplear ya que de cara a realizar el diseño del PCB condicionará la selección de unos u otros.

4. DISEÑO DE LA PLATAFORMA

Una vez estudiado el tipo de conversor a emplear, analizado como está implementado y conocida la plataforma Arduino, se va a proceder a explicar cuál ha sido el proceso de diseño de la plataforma. Dicha plataforma va a estar formada por el integrado que contiene la parte analógica del CAD, la cual irá en un PCB junto con su circuitería asociada, y la plataforma Arduino. El PCB irá incorporado a modo de shield sobre Arduino.

Primero se va a mostrar el proceso de diseño del PCB, explicando las peculiaridades del circuito así como los pasos que se han seguido en su desarrollo. Posteriormente se van a explicar los algoritmos realizados, indicando como se han implementado así como algunos detalles del código como la configuración de los timers.

4.1. DISEÑO DEL SHIELD

Como se ha citado en la introducción del proyecto, un **shield** se trata de un módulo que posee una funcionalidad determinada el cual se puede acoplar a la placa Arduino mediante los diferentes puertos que esta posee. En el mercado existen multitud de shields o módulos Arduino tales como módulos Ethernet, Wi – Fi, tarjetas SD, control de motores, etc. Dichos módulos también siguen la filosofía de hardware libre y se pueden consultar sus esquemáticos sin ningún problema en la red. En la siguiente figura se muestra el ejemplo de un shield para Arduino, correspondiente a un módulo Ethernet.

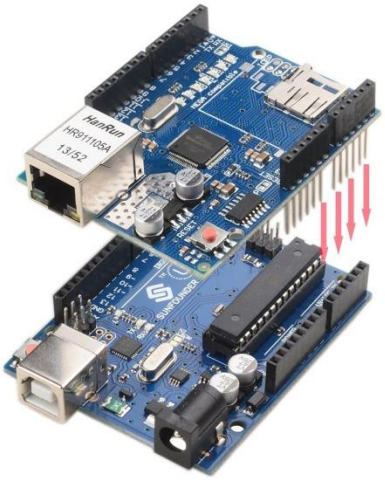


Figura 40: Ejemplo de un shield para Arduino

Para poder desarrollar la plataforma de implementación y testeo del CAD, se va a diseñar un PCB que actuará a modo de shield para Arduino. En dicho PCB se va a incluir el integrado con la parte analógica del conversor así como los componentes externos que precise tales como resistencias, condensadores y la referencia de tensión. Se le incorporarán una serie de jumpers para conectarlo a los diferentes puertos que posee Arduino.

El software de diseño empleado para el diseño del PCB es **Orcad Cadence**. En él se realizará el diseño del esquemático, la creación de los footprints necesarios, compilación del

mismo y finalmente el trazado de las pistas. Una vez terminado el diseño del PCB se generarán una serie de archivos llamados gerbers, que contendrán toda la información necesaria para fabricar la placa (propiedades de las pistas, anchuras de los agujeros de los taladros, etc.).

4.1.1. ALIMENTACIÓN DEL SHIELD

En el capítulo 3 se mencionó el método de alimentación del conversor. Dado que Arduino trabaja con valores comprendidos entre 0V y 3.3V, estas serán las tensiones de alimentación del integrado. Por lo tanto, se empleará una alimentación asimétrica. Esto implicará que la tensión de referencia del integrado o masa no pueda ser 0V, sino que tiene ser una tensión que será la mitad del rango de tensiones de alimentación de este. Por lo tanto, la tensión de referencia del integrado, a la que denominaremos "**tierra analógica**" o **Vta**, será:

$$V_{ta} = \frac{V_{dd} + V_{ss}}{2} = \frac{3.3V + 0V}{2} = 1.65V$$

En el caso de haber empleado una alimentación simétrica, las tensiones de alimentación serán 3.3V y -3.3V, estando la referencia del integrado o masa a 0V.

En la siguiente tabla se resumen los valores de tensión de alimentación, referenciados a los correspondientes pines del integrado.

Tensión de alimentación	Valor	Pin
Vdd	3.3V	40
Vss	0V	21
Vta	1.65V	8, 13 y 34

Tabla 8: Valores de tensión de alimentación

No se debe confundir la tierra analógica del integrado con la masa de Arduino. Son dos referencias completamente diferentes. La necesidad de la tierra analógica radica en que el integrado posee una serie de pines que están conectados a masa, pero como la tensión de alimentación Vss es 0V, no podemos alimentar ambos a 0V. Por ese motivo se debe conectar la tierra analógica al valor mitad de la tensión de alimentación. En cualquier circuito electrónico la tensión de referencia de este la podemos conectar al valor que nos interese, siendo normalmente el valor mitad de las tensiones de alimentación.

Para alimentar los pines del integrado Vdd (Pin 40) y Vss (Pin 21) del integrado, se van a conectar a Arduino Due. Sabemos que este nos proporciona una serie de tensiones de referencia como 3.3V y 0V o GND. Por lo tanto nos aprovecharemos de ellas para alimentar al integrado. Dado que la corriente máxima que nos proporciona la salida de 3.3V de Arduino Due es de 800mA y el integrado tiene un consumo de 20mA, no existirá ningún problema en cuanto a requerimientos de corriente.

Para obtener la tensión de 1.65V, se va a emplear una referencia de tensión de la casa Texas Instruments®. El modelo a utilizar será el **REF1933**. Dicho integrado nos proporciona dos tensiones estables de 3.3V y 1.65V, siendo la de 1.65V la que nos va a interesar para nuestra aplicación. La tensión de alimentación del mismo será de 5V, la cual se obtendrá de la salida de referencia de Arduino Due de 5V. También debe estar conectado a una tensión de 0V (GND) la cual se obtendrá de Arduino. Dicho dispositivo está optimizado para aplicaciones de procesado digital de señal, sistemas de adquisición de datos así como sistema que procesan una tensión de alimentación unipolar (*Single – Supply Systems*).

En la siguiente figura se muestra el encapsulado del integrado REF1933, y en la tabla la relación de pines del mismo.

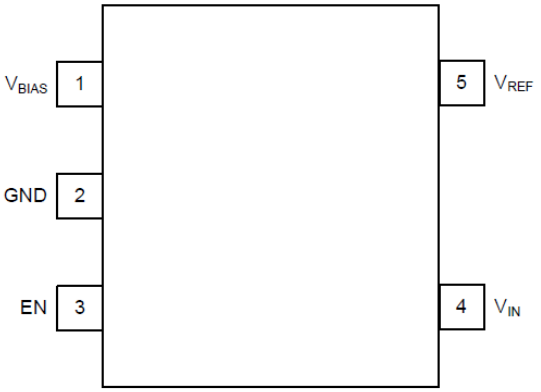


Figura 41: Encapsulado SOT23_5 de la referencia de tensión REF1933

Pin		E/S	Descripción
Nombre	Nº.		
Vbias	1	S	Voltaje de salida Vbias; Valor (Vref / 2) = 1.65V
GND	2	-	Masa del integrado
EN	3	E	Habilitación del integrado (con EN ≥ Vin - 0.7V se habilita)
Vin	4	E	Alimentación del integrado, a 5V
Vref	5	S	Voltaje de salida Vref; Valor (Vref) = 3.3V

Tabla 9: Descripción de los pines de la referencia de tensión REF1933

La corriente que proporciona cada una de las salidas de la referencia de tensión (Vref y Vbias) es de 360µA. Dado que están conectadas a las entradas no inversoras de los amplificadores (pines 8 y 13) y al pin de referencia del integrado (pin 34) no habrá ningún problema en cuanto a requerimientos de corriente.

La tecnología de fabricación de la referencia de tensión es **SMD** con un encapsulado SOT23 – 5. De cara a realizar su implementación en el PCB, el fabricante recomienda conectar unos condensadores cerámicos de baja resistencia serie o ESR de 0.1µF en Vin, Vbias y Vref para reducir y desacoplar posibles ruidos. En la siguiente figura se muestra la idea.

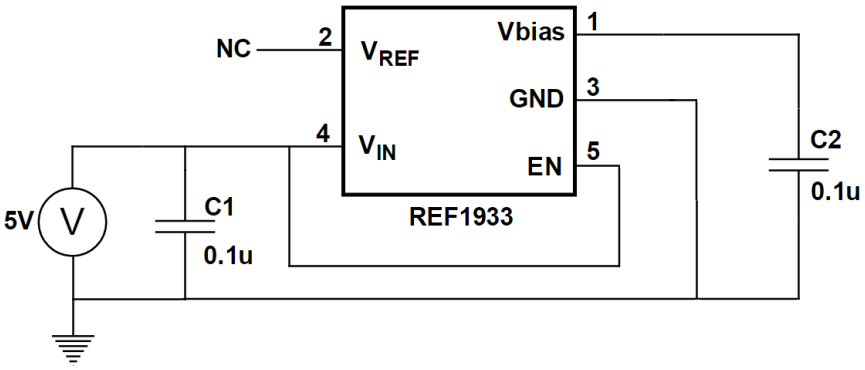


Figura 42: Esquema del circuito de alimentación

El pin EN (Enable) se va a conectar a Vin para que se encuentre habilitado permanentemente. Dado que la salida Vref no se va a emplear, no se conectará el condensador y se dejará no conectada. En la siguiente tabla se resume la relación de pines entre la referencia de tensión, Arduino y el integrado.

Pin de la Referencia de Tensión	Pin de Arduino Due	Pines del Integrado
Vbias	-	Pines 8, 13 y 34
GND	Pin GND	Vss
EN	Pin 5V	-
Vin	Pin 5V	-
Vref	-	-

Tabla 10: Relación de pines entre la referencia de tensión, Arduino y el integrado

Finalmente, se muestra el esquema del circuito de alimentación, en el cual se indica la referencia de tensión, la alimentación del mismo mediante Arduino y las salidas de este conectadas al integrado con el conversor.

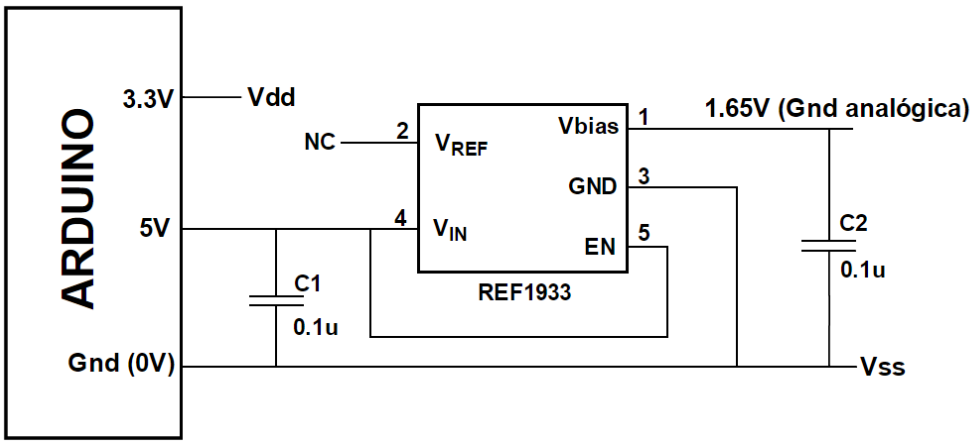


Figura 43: Esquema de alimentación de la plataforma

4.1.2. INTEGRACIÓN DEL CONVERSOR A/D EN ARDUINO DUE

Una vez se ha alimentado el integrado de forma correcta, se va a proceder a explicar cómo se va a controlar este mediante Arduino. En el apartado 3.2.3. se han mencionado los pines del integrado que deben ser controlados por Arduino, pero no se ha comentado que pines de Arduino serán.

En la siguiente figura se muestra la conexión de los pines del integrado con los puertos de Arduino.

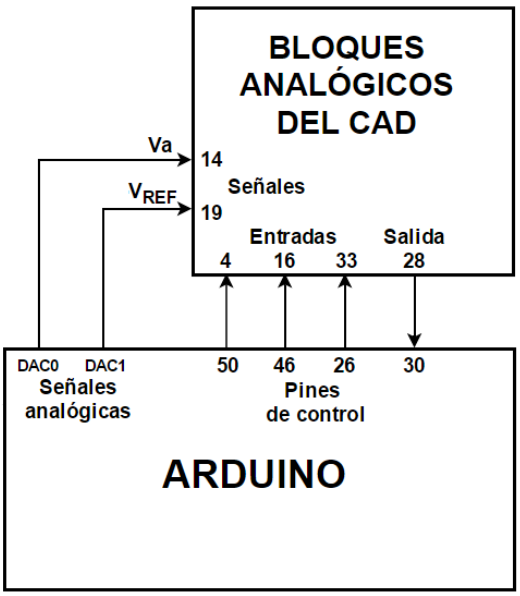


Figura 44: Conexión de los puertos de Arduino junto con el integrado

Los pines 4 (CLIpn), 16 (CLIRVin), 28 (Out) y 33 (IMCLKR) se van a conectar a los puertos digitales de Arduino. Por el contrario, los pines 14(Vin+) y 19(Vref+) se conectarán a las salidas analógicas o CDA. De momento no se va a proceder a la explicación sobre la configuración de los puertos ya que se realizará más adelante. En la siguiente tabla se resumen los pines del integrado que van a ser controlados por Arduino. La elección de los pines se debe a las características de diseño del PCB.

Pin del Integrado	Pin de Arduino Due
4	50
14	DAC0
16	46
19	DAC1
28	30
33	26

Tabla 11: Resumen los pines del integrado que van a ser controlados por Arduino

4.1.3. ELEMENTOS ADICIONALES DEL INTEGRADO

El integrado, además de ser controlado y alimentado por Arduino así como por la referencia de tensión, precisa de una serie de hardware adicional para su funcionamiento. En la figura 45 se pueden observar dichos elementos. Aunque ya se han mencionado en el apartado 3.2., se va a realizar un estudio más detallado sobre ellos.

En la siguiente figura se muestra el integrado junto a los componentes externos que necesita. Para comprender mejor la estructura interna del integrado, acudir a la figura 30 y 31.

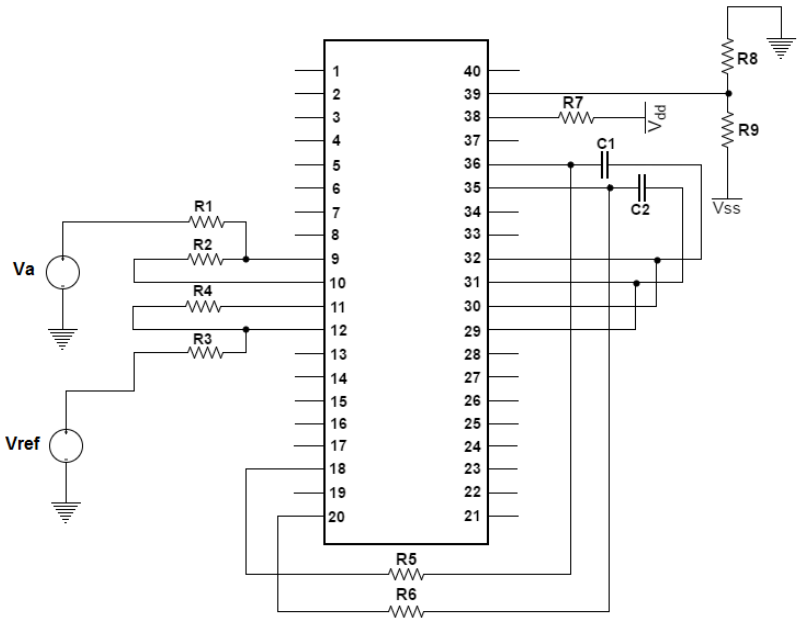


Figura 45: Integrado junto con sus componentes externos

- Las resistencias R1y R2, R3 y R4 controlan las ganancias de los amplificadores en configuración inversora. Como están en configuración inversora con ganancia unidad, se cumplirá que las cuatros resistencias tendrán el mismo valor.
- La resistencia R5 y el condensador C1 controlan la pendiente de integración. Lo mismo ocurre para la resistencia R6 y el condensador C2. Dado que estamos manejando señales diferenciales en el procesado, se requiere que R5 y R6 así como C1 y C2 sean iguales. En el caso de los condensadores es un factor muy importante, ya que suelen tener valores de tolerancias mayores que las resistencias y por tanto dar lugar a valores ligeramente diferentes entre dos condensadores iguales. Por ese motivo, cuando se realice la selección de los mismos se medirán con un medidor de impedancias con el fin de seleccionar dos condensadores que sean los más parecidos posibles. Otra alternativa sería comprar condensadores de mayor precisión pero por el contrario presentan un precio superior respecto a los condensadores estándar.
- Las resistencias R7, R8 y R9 se encargan de polarizar las fuentes de corriente del integrado. Aunque no se ha analizado la arquitectura microelectrónica del integrado, las diferentes etapas que lo forman emplean una serie de fuentes de corriente. La

polarización de las fuentes de corriente se realiza con las tres resistencias anteriormente citadas. A través de R8 y R9 se consigue un divisor de tensión que polariza la puerta del transistor, en configuración cascodo. Finalmente, mediante R7 se obtiene la corriente que se replicará en las diferentes etapas del integrado.

4.1.4. PROCESO DE DISEÑO DEL SHIELD

Una vez analizado el funcionamiento del integrado y los elementos externos necesarios para su funcionamiento tales como la referencia de tensión o las resistencias y condensadores, el paso final será la integración de todos ellos en un PCB. Como se ha citado anteriormente, dicho PCB irá montado sobre Arduino a modo de **shield**. Este poseerá una serie de pines de conexión que irán conectados a las diferentes entradas y/o salidas de Arduino.

En la siguiente figura se muestra el esquema de conexionado con los bloques de la plataforma, especificando los puertos del integrado y de Arduino.

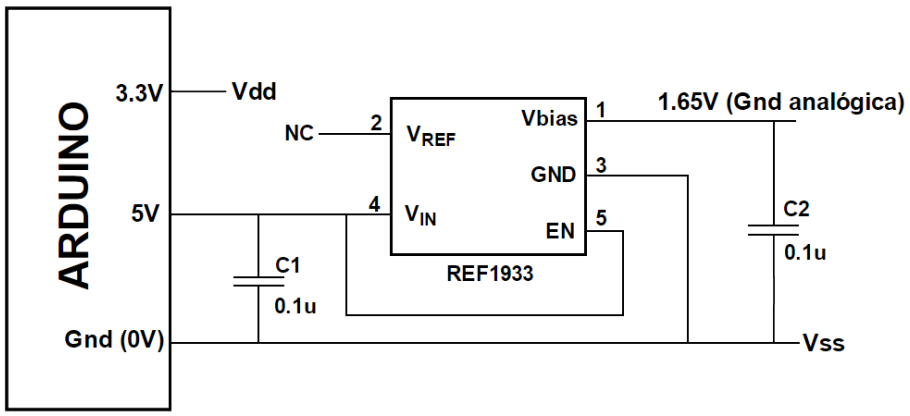


Figura 46: Conexionado de Arduino, el integrado y la referencia de tensión

El PCB estará formado por los siguientes elementos:

- El integrado
- Las resistencias y condensadores que configuran al integrado
- La referencia de tensión junto con los condensadores de filtrado
- Jumpers de conexionado

Antes de iniciar el diseño del PCB, se deben tener en cuenta una serie de consideraciones en el desarrollo. La más importante de todas es la localización de los diferentes puertos de Arduino, los cuales condicionarán la ubicación de los jumpers del shield. En la siguiente figura se muestra la placa Arduino, en la cual se indican los diferentes puertos que esta posee. Se marcan en negrita los pines de Arduino que se van a conectar con el PCB. Por lo tanto, es importante la ubicación de estos ya que la colocación de los jumpers en el shield estarán condicionados por estos. Además, con una adecuada colocación del integrado se facilitará el trazado de las pistas.

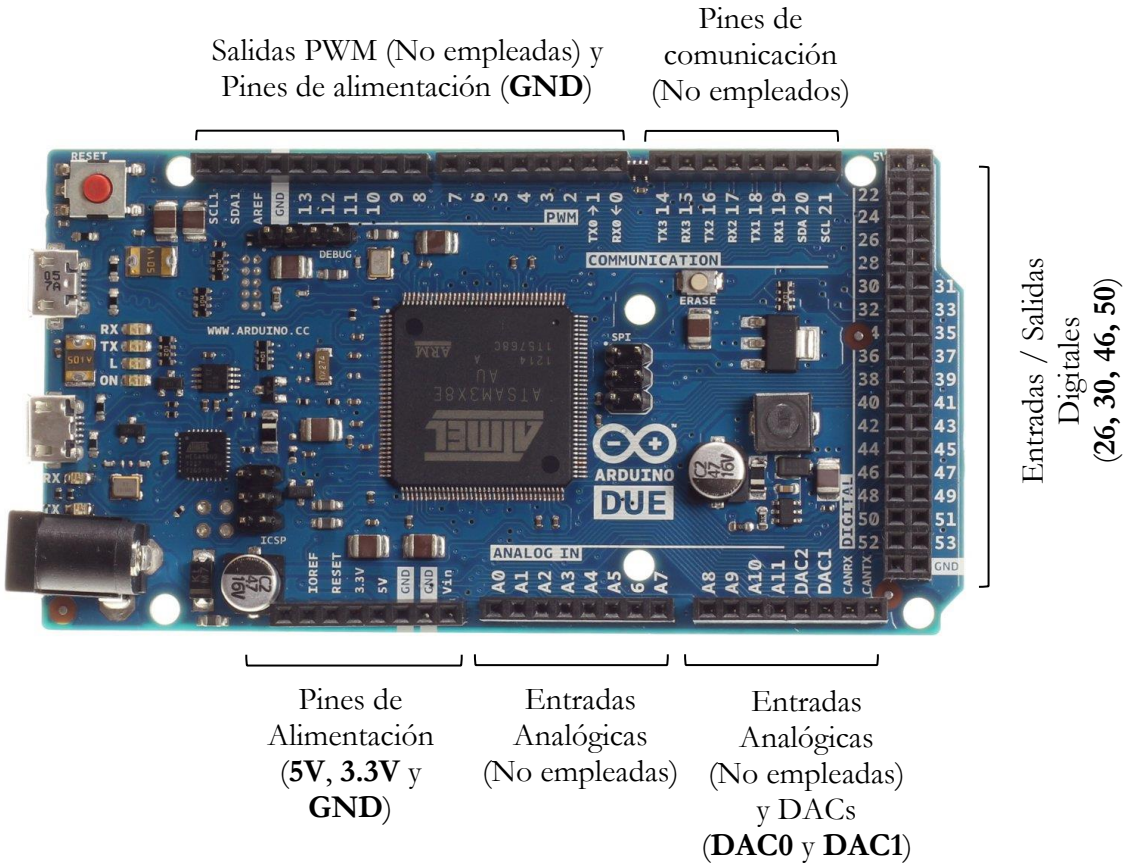


Figura 47: Puertos de la placa Arduino Due

Teniendo en cuenta el criterio anterior, se muestra en la siguiente imagen el esquema del shield, en el cual se ha considerado la localización de los puertos de Arduino y la posición del integrado.

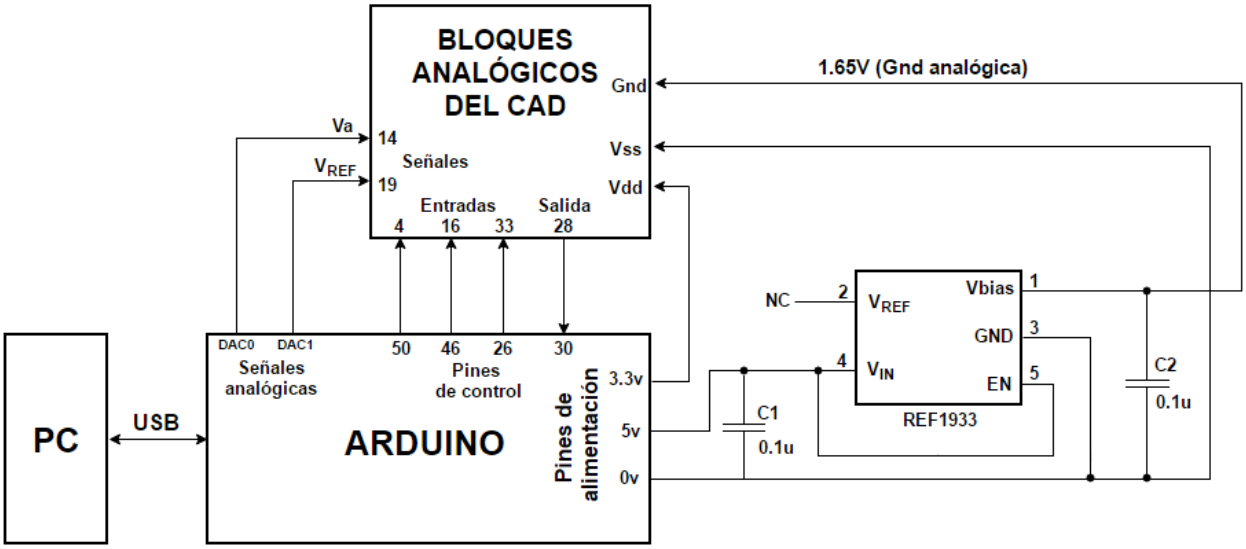


Figura 48: Esquema del shield

A partir de este esquema, ya se puede iniciar el proceso de diseño del PCB. El software empleado es el **OrCAD Capture**, de **Cadence**. En él se va a realizar el diseño del esquemático, la asignación y creación de los footprints (en los casos que sea necesario) y el trazado de las pistas del PCB. Por último se crearán una serie de archivos que servirán para fabricar la placa.

El primer paso para la elaboración del PCB es crear un proyecto en OrCAD. En el crearemos el esquemático de la figura 48. Hay algunos componentes que no están en las librerías de OrCAD por lo que se deben crear de manera manual. Estos componentes son el **conversor A/D** y la **referencia de tensión REF1933**. De cara a crear dichos componentes, deberemos especificar sus entradas, salidas y el tipo de estas (alimentación, pasivas, etc.). Cuando se tengan todos los componentes necesarios para el esquemático, el siguiente paso será elaborar el esquema completo. En el anexo 2 se muestra el esquemático que se ha elaborado en OrCAD donde se detallan todos los componentes empleados, sus referencias, etc. De cara a futuros apartados en los que se necesite referenciar a los componentes del esquemático, emplearemos las referencias del anexo 2.

A cada uno de los componentes del esquemático se debe asignar un **footprint**. El footprint consiste en la representación de los pads de un componente, usado para realizar los agujeros correspondientes en el PCB y sobre el cual irá el componente. En OrCAD existen footprints que se encuentran en librerías por defecto, pero habrá otros que habrá que crearlos desde cero. Cuando se cree un footprint, habrá que especificar el tipo de encapsulado, el tamaño de los pads, etc., cuyas características se deberán consultar en el datasheet del componente. En el editor aparecerán los pads numerados, y deben coincidir con los números de los pines de los componentes empleados en el esquemático. Si no puede dar lugar a errores en el trazado de las pistas que impliquen que haya que volver a repetir el diseño.

En la siguiente tabla se muestra un resumen de los componentes empleados el esquemático, junto con sus respectivos footprints. También se especifica la tecnología de cada componente, ya que algunos serán **discretos** y otros de tipo **SMD**. Las referencias de cada componente se pueden consultar en el anexo 2. Se marcarán en rojo los componentes que han sido creados desde cero, y en azul los footprints.

Componente	Valor	Footprint	Tecnología
ADConverter		dip40_6	Discreto
REF1933		sot23_5	SMD
R1	100kΩ	res400	Discreto
R2	100kΩ	res400	Discreto
R3	100kΩ	res400	Discreto
R4	100kΩ	res400	Discreto
R5	100kΩ	res400	Discreto
R6	100kΩ	res400	Discreto
R7	150kΩ	res400	Discreto
R8	100kΩ	res400	Discreto

R9	100kΩ	res400	Discreto
C1	1nF	cap300	Discreto
C2	1nF	cap300	Discreto
C3	0.1µF	smdcap_1206	SMD
C4	0.1µF	smdcap_1206	SMD
J1		jumper8_recto	Discreto
J2		jumper8_recto	Discreto
J3		jumper8_recto	Discreto
J4		jumper18_recto	Discreto
J5		jumper8_recto	Discreto
J6		jumper8_recto	Discreto
J7		jumper10_recto	Discreto

Tabla 12: Resumen de los componentes y footprints empleados en el diseño del shield

A continuación se comentan algunas consideraciones sobre los componentes:

- El conversor emplea un encapsulado **DIP40**, cuyas medidas están estandarizadas.
- La referencia de tensión emplea un encapsulado **SOT23_5**, cuyas medidas están estandarizadas.
- Las resistencias empleadas son de **1/4 de vatio**.
- Los condensadores discretos son de **tantalio**, cuyas medidas están estandarizadas. Se debe procurar que los dos condensadores poseen la misma capacidad, debido al procesado de las señales diferenciales.
- Los condensadores SMD son de **baja ESR**. Su encapsulado es de **métrica 1206**, cuyas medidas están estandarizadas.
- Los jumpers también presentan medidas estandarizadas. Para calcular la longitud de ellos, se debe multiplicar el **número de pines del mismo por 25.4mm** (1mil Pulgada).

En la siguiente figura se muestran los footprints empleados en el diseño.

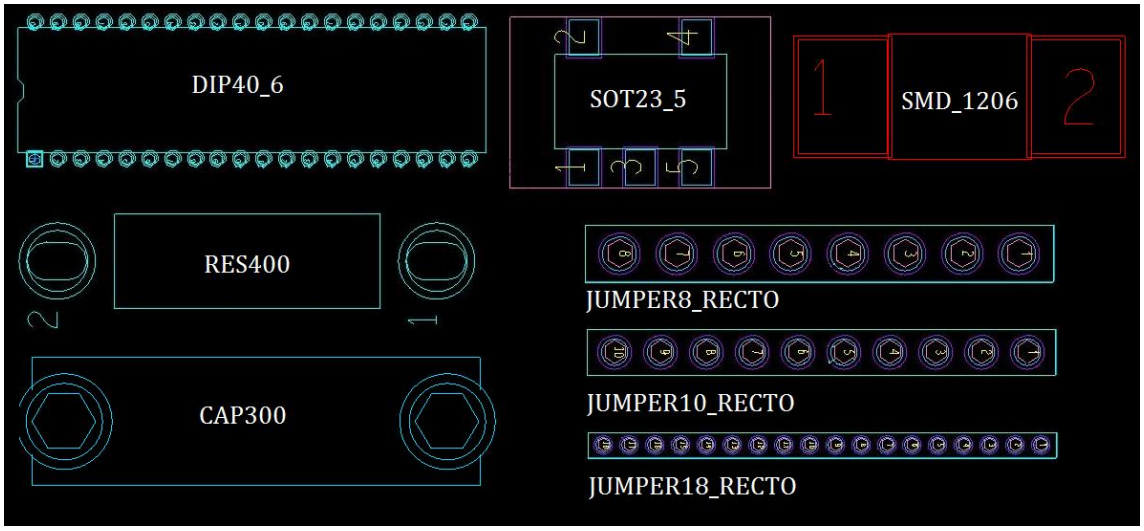


Figura 49: Imágenes de los footprints empleados

Una vez asignados los footprints ya se puede pasar a la realización del PCB. Para ello se comprueba que no hay errores en el diseño **DRC**, se comprueba la **lista de los materiales** (*Bill of materials*) y se realiza el **netlist**. Con el netlist se tienen las referencias de todos los componentes junto con las conexiones entre ellos. A partir de él se realizará el diseño del PCB.

Una vez se haya compilado el esquemático, mediante el paquete **PCB Editor** de OrCAD. Capture se realizará el trazado de las pistas. Cabe destacar que en el diseño se va a realizar con componentes discretos y SMD. Esto será muy importante de cara a plantear las caras top y bottom de la placa ya que la colocación incorrecta de una pista puede dar lugar a errores en el diseño. Es importante remarcar que la placa debe tener unas dimensiones determinadas para que encaje encima del Arduino. Para el proyecto se han elegido **95mm (ancho) X 65mm (alto).**

Se van a proponer una serie de pautas para la realización del PCB que se deben tener en cuenta para evitar posibles errores en el diseño:

- Anchura de pistas entre **12mils** y **24mils**. La anchura óptima es de 24mils aunque en aquellos casos en los que sea difícil el trazado de estas se pueden dejar a 12mils. Estas anchuras se deben tener en cuenta para la fabricación en el laboratorio de la Universidad Pública de Navarra. Si se enviaría a fabricar fuera habría que tener en cuenta los criterios que marcaría la empresa correspondiente.
- Realizar las conexiones y el trazado de pistas en la capa **bottom** para los componentes **discretos** ya que estos se deben soldar en esta. En algunos casos se podrán emplear vías para pasar la pista a la capa top Una **vía** es un agujero a modo de pad que permite que una pista pase de una capa a otra. De forma general, trazar las pistas y soldar los componentes discretos en la cara opuesta a la que se colocan dichos componentes.
- Realizar las conexiones y el trazado de pistas en la capa **top** para los componentes **SMD**. De forma general, trazar las pistas y soldar los componentes discretos en la misma cara en la que se colocan dichos componentes.
- Para el caso de los **jumpers**, estos irán acoplados por debajo de la capa top, pero los pines se deben soldar en esta.
- Procurar hacer las pistas lo más **cortas** posibles para evitar el acoplo del ruido en estas.
- Eliminar las **islas** de cobre de las capas top y bottom para evitar problemas de acoplo capacitivo.
- Tratar de emplear el **mínimo** número de vías.
- Revisar con detalle el diseño antes de enviarlo a fabricarlo.
- Tener paciencia, cuidado y gusto en la soldadura de los componentes SMD, especialmente en el caso de la referencia de tensión. Se puede usar un microscopio para facilitar la colocación del componente

La conclusión de estas pautas surge de la experiencia del diseño del mismo así como de otros anteriores, en los que no se tuvieron en cuenta dando lugar a errores que hubo que solventarlos, principalmente con la colocación de puentes mediante cables físicos sustituyendo a las pistas.

En las siguientes figuras se muestran las imágenes de las capas top y bottom del diseño. Retomando las medidas de la placa, esta debe tener 95mm (ancho) X 65mm (alto).

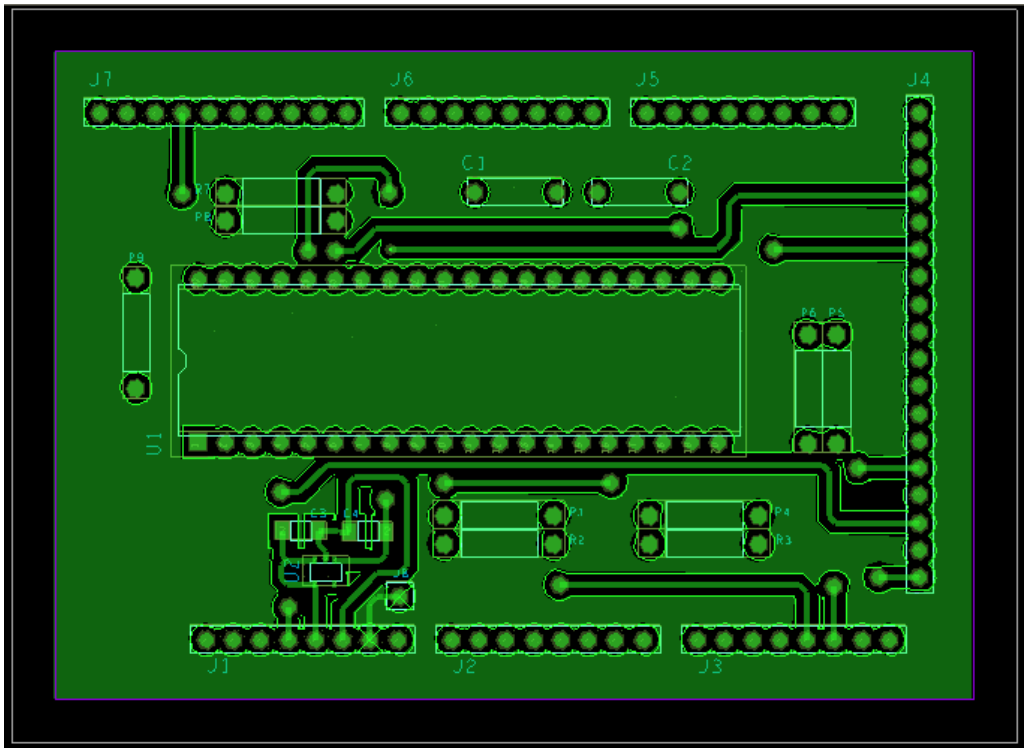


Figura 50: Capa top del shield

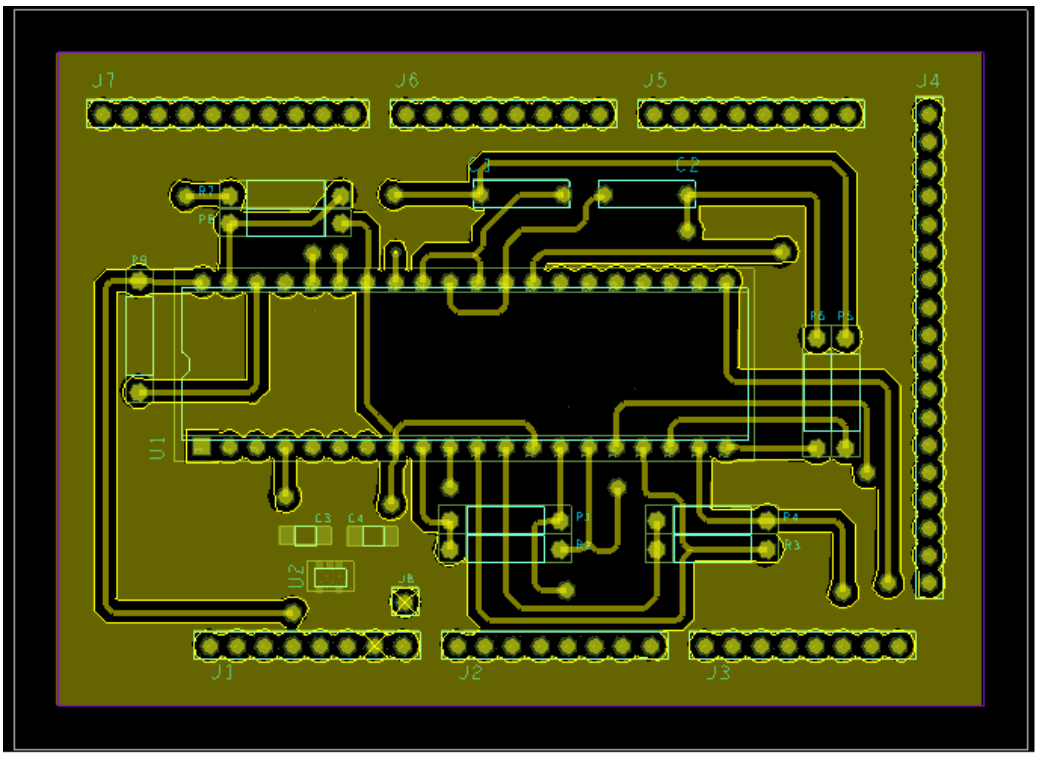


Figura 51: Capa bottom del shield

Una vez se haya finalizado el diseño del PCB, faltarían crear los archivos para su fabricación. Los archivos necesarios son los siguientes:

- **Top.art**, el cual contiene información sobre la cara Top.
- **Bottom.art**, el cual contiene información sobre la cara Bottom.
- **Boardoutline.art**, el cual contiene información sobre el contorno de la placa.
- **Gerger.drill**, el cual contiene información de los diferentes agujeros del PCB para realizar el taladrado de los mismos.

Con dichos archivos se tendrá toda la información necesaria para fabricar el PCB. Finalmente se muestra en la siguiente figura el resultado final del PCB.

Una vez se tiene el shield, faltaría testear su funcionamiento. Para ello se deben comprobar las tensiones de alimentación, y realizar algún pequeño algoritmo en Arduino para comprobar que integra y compara correctamente. En el apartado de resultados se comentarán dichos aspectos.

Tras haber testeado la placa, faltaría desarrollar el sistema final: la elaboración de los diferentes algoritmos de conversión. En el apartado siguiente se van a detallar los diferentes algoritmos que se han realizado, explicando las peculiaridades de cada uno así como la configuración de los mismos.

4.2. ALGORITMOS

En este apartado se van a explicar los diferentes algoritmos de conversión desarrollados. Una vez se haya fabricado la placa, soldado los diferentes componentes y testeado su funcionamiento se va a proceder al desarrollo de la parte digital del conversor. Como ya se ha citado, la funcionalidad de Arduino es controlar al integrado que posee la parte analógica del conversor. Como este va a ir integrado en un PCB con los diferentes componentes y montado sobre Arduino a modo de shield, mediante una adecuada programación se controlará al mismo. La idea es sustituir la parte digital que aparece en la figura 26 por software. De esta manera controlaremos el proceso de conversión seleccionando la tensión de entrada analógica o la de referencia, invirtiendo la señal o controlando los tiempos de integración y comparación con los timers.

En el primer apartado se hará una mención al método de configuración de los algoritmos, indicando como configurar los timers, los puertos, etc. En los siguientes apartados se desarrollarán los algoritmos basándonos en el método de configuración y apoyándonos en diagramas de flujo para comprender su funcionamiento. Se han desarrollado tres algoritmos: uno básico el cual se corresponde con la conversión lineal de la señal, explicada en el apartado 2.4.4; otro básico encargado de la conversión no lineal de la señal, pero el cual no empleará el contador no lineal mencionado, sino que empleará una tabla que posee los valores ya linealizados; y por último uno avanzado en el que si se empleará el contador no lineal. Este se corresponde con el diseño final de la plataforma que da nombre al título del presente proyecto.

4.1.1. CONFIGURACIÓN DE LOS ALGORITMOS

La configuración de Arduino es esencial para un correcto funcionamiento de la plataforma. A continuación se va a mostrar los diferentes métodos de configuración de cada uno de los elementos del sketch. Dado que el IDE de Arduino emplea una sintaxis propia, se va comentar los comandos más importantes que sean útiles para el desarrollo de los algoritmos. De esta manera se entenderá de una manera más clara los diferentes sketches.

COMPOSICIÓN DEL SKETCH

Como se ha citado en el apartado 3.2.1., el sketch de Arduino se compone de dos partes:

- El **setup ( )** en el cual se realiza la configuración de puertos, timers, etc. Dicha parte se ejecutará sólo una vez.
- El **loop ( )** en el cual se ejecuta la lógica de programa. Dicha parte se ejecutará repetidamente, o como indica su propio nombre, en un bucle.

La declaración de variables se puede realizar tanto dentro como fuera del setup ( ). En el caso de declararlas fuera del se deben declarar antes que este.

A continuación se muestra un pequeño trozo del sketch que posee ambas partes.

```
/* Declaración de variables. */

void setup() {
  /*  Código de configuración, el cual se ejecuta una sola vez. */
}

void loop() {
  /* Lógica de programa, ejecutada repetidamente. */
}
```

Una vez tengamos esta estructura se puede proceder a la programación del sketch.

CONFIGURACIÓN DE LOS PUERTOS DIGITALES

Para el control de los diferentes pines de control del integrado (4, 16 y 33) así como la salida que proporciona el resultado de la comparación (28) se van a emplear los puertos digitales de Arduino.

En la siguiente tabla se resumen los pines del integrado que van a ser controlados por Arduino, y el tipo de configuración que van a tener. La configuración es evidente: los de tipo salida servirán para activar ciertos pines, y la entrada obtendrá el resultado de la comparación que indicará el fin de la conversión. Para más obtener más información sobre los pines del integrado consultar la tabla 4 del apartado 3.2.

Pin del Integrado	Pin de Arduino Due	E/S del pin de Arduino
4	50	Salida
16	46	Salida
28	30	Entrada
33	26	Salida

Tabla 13: Resumen de los pines del integrado

Para configurar los pines de Arduino se va seguir el siguiente procedimiento:

- Antes y fuera del setup ( ) o dentro del mismo, declaramos las variables que se van a corresponder con los puertos de Arduino. Serán **constantes** ya que no van a variar a lo largo del programa y de tipo **entero** ya que representarán el número del puerto.

```
/* Pines de control del CAD */
const int IMCLKR = 26;   // Reseteo del contador
const int CLIRVin = 46;   // Conmuta de Vin a Vref el conversor
const int CLIpn = 50;      // Invierte la señal de entrada

/* Pin de salida de CAD */
const int OUT = 30;          // Detecta el resultado de la comparación
```

- Dentro del setup ( ) se realiza la configuración de los puertos.

```
/* Configuración del puerto como entrada o salida */
pinMode(IMCLKR, OUTPUT);
pinMode(CLIRVin, OUTPUT);
pinMode(CLIpn, OUTPUT);
pinMode(OUT, INPUT);
```

Mediante la directiva **OUTPUT** se configurarán como salida y mediante **INPUT** como entrada. Para los pines de salida, se les puede asignar un valor alto o bajo a través del siguiente comando.

```
/* Escribimos el valor digital en el puerto */
digitalWrite(CLIRVin, HIGH);
digitalWrite(IMCLKR, LOW);
```

Se seleccionará el pin que queremos configurar, y mediante **HIGH** o **LOW** le asignaremos el valor lógico.

CONFIGURACIÓN DEL LOS PUETOS ANALÓGICOS: DACs

Los DACs de Arduino serán indispensables ya que con ellos generaremos las señales analógicas de entrada del conversor así como la tensión de referencia. Presentan una resolución entre 8 y 12 bits, pudiendo configurar el rango disponible entre ambos valores (8, 9, 10, 11 y 12 bits). La configuración los mismos se realizarán dentro del setup ( ). A continuación se muestra el método de configuración.

```
/* Configuración de la resolución del DAC */
analogWriteResolution(10);  // Seleccionamos la resolución del DAC, en este caso de 10 bits

/* Escribimos el valor en el DAC correspondiente */
analogWrite(DAC0,analogValue);  // Escribimos la tensión analógica
analogWrite (DAC1,ref);         // Escribimos la referencia de voltaje
```

En el primer comando escribimos la resolución que se va a emplear en los DAC. El valor que se escriba será el mismo para ambos. En el segundo comando escribiremos el valor analógico en el DAC correspondiente. Mediante DAC0 o DAC1 seleccionaremos el DAC correspondiente, y mediante analogValue o ref se escribirá el valor analógico en dicha salida. El valor de dichas variables se deben declarar previamente a este comando, indicando el valor que queramos y teniendo en cuanta la resolución seleccionada ($2^N - 1$, con N la resolución del DAC).

Un aspecto a tener en cuenta sobre los DAC es que introducen un **offset de DC en DAC0 y DAC1**, es decir, la salida de los DACs presentan un pequeño offset de 550mV. Dicho offset se puede eliminar con un **condensador de desacoplo**. Por otra parte, el **rango del DAC** va entre 1/6 ADVREF y 5/6 ADVREF, donde ADVREF es 3.3V.

CONFIGURACIÓN DE LOS TIMERS

Tal como se ha ido citando a lo largo de los anteriores apartados, los **timers** van a ser el elemento clave en el desarrollo de los algoritmos. Se tratan de funciones que se ejecutan cada un determinado número de ciclos de reloj. Ese número de ciclos de reloj equivaldrán a un tiempo determinado, de esta manera se puede realizar llamadas a funciones controlando los tiempos de una manera muy precisa. El microcontrolador posee **nueve timers** que pueden ser configurados y usados de manera independiente. Están organizados en tres bloques (**TC0, TC1 y TC2**), y cada uno posee a su vez tres canales (**0, 1 y 2**).

Para controlar los tiempos de llamada, Arduino dispone de varias frecuencias de reloj, las cuales se obtienen dividiendo el Master Clock (MCK, de 84MHz) por una serie de divisores de frecuencia (2, 8, 32, 128 y SCLK), obteniendo cinco frecuencias diferentes:

- **TIMER_CLOCK1**: MCK / 2 = 42 MHz
- **TIMER_CLOCK2**: MCK / 8 = 10.5 MHz
- **TIMER_CLOCK3**: MCK / 32 = 2,625 MHz
- **TIMER_CLOCK4**: MCK / 128 = 656,250 KHz
- **TIMER_CLOCK5**: SLCK = 32768 KHz

Para calcular el número de ciclos de reloj necesarios para un tiempo determinado, se empleará la siguiente ecuación:

$$nº.\,de\ ciclos = \ tiempo(s) * TIMER\_CLOCKx\ (MHz),\ \ con\ x = 1,2,3,4\ o\ 5$$

Por ejemplo, para un tiempo de 1mS, si se emplearía el TIMER_CLOCK4 de 656,250KHz, saldrían un total de 1mS · 656,250 KHz = 657 ciclos de reloj. De esta manera, se obtendría el número de ciclos de reloj necesarios para un tiempo determinado.

Para configurar los timers, se va a realizar el siguiente procedimiento **dentro del set up( )** :

- Deshabilitamos la protección contra escritura de los **registros PMC** (*Power Management Controlers*). Por defecto están desactivados, debido a que el consumo de potencia es un factor importante para determinadas aplicaciones.

pmc_set_writeprotect(false); // Desabilitamos la protección de los registros PMC

- Una vez deshabilitados los registros PMC, se habilitan las entradas de reloj externas.

pmc_enable_periph_clk(ID_TC3); // Habilitamos la entrada de reloj externa TC3

Como parámetro de entrada seleccionaremos el timer que se quiere emplear. Dado que existen nueve timers, se seleccionará el timer con el parámetro **ID_TCx**, con x = 1, 2, 3, 4, 5, 6, 7, 8 o 9 representando uno de los nueve timers disponibles.

- Con el timer seleccionado, se va a proceder a su configuración. Primero seleccionamos uno de los tres bloques disponibles (TC0, 1 o 2) así como el canal (0, 1 o 2). También seleccionaremos el modo (forma de onda o medida) así como otros parámetros.

TC_Configure(TC1 /* Clock */,0 /* Canal */,
TC_CMR_WAVE /* Modo forma de onda */ |
TC_CMR_WAVSEL_UP_RC /* Disparo del Timer por comparación con los clicks de reloj */ |
TC_CMR_TCCLKS_TIMER_CLOCK4); /* Selección del divisor del máster clock */

En el parámetro **clock**, seleccionamos uno de los tres bloques y en **canal** uno de los tres canales disponibles. El resto de parámetros indican el modo del timer. Se empleará el modo forma de onda **TC_CMR_WAVE** ya que se van a generar señales digitales para controlar los pines del integrado. Mediante **TC_CMR_WAVSEL_UP_RC** se indica que cuando se cumpla el número de ciclos de reloj se disparará la función. Por último, **TC_CMR_TCCLKS_TIMER_CLOCK4** selecciona una de las cinco frecuencias de reloj anteriormente citadas.

En la siguiente tabla se resumen los bloques y canales disponibles en Arduino Due, así como la entrada de reloj de cada uno de ellos.

Instancia	TC	Canal	Entrada de Reloj Externa
T0	TC0	0	TCLK0
T1	TC0	1	TCLK1
T2	TC0	2	TCLK2
T3	TC1	0	TCLK3
T4	TC1	1	TCLK4
T5	TC1	2	TCLK5
T6	TC2	0	TCLK6
T7	TC2	1	TCLK7
T8	TC2	2	TCLK8

Tabla 14: Resumen de los bloques y canales disponibles en Arduino Due

Seguidamente, se configura timer el número de ciclos de reloj que necesita para su disparo.

TC_SetRC(TC1, 0, 657); //Configuramos el número de ciclos de reloj que harán disparar al Timer

Habilitamos las interrupciones de los tiemers modificando los siguientes registros:

TC1->TC_CHANNEL[0].TC_IER=TC_IER_CPCS; // IER = interrupt enable register
TC1->TC_CHANNEL[0].TC_IDR=~TC_IER_CPCS; // IDR = interrupt disable register

Se deberá seleccionar el bloque así como su canal para cada interrupción que se habilite.

- Finalmente, activamos el vector que maneja las interrupciones.

NVIC_EnableIRQ(TC3_IRQn);

Como parámetro de entrada seleccionaremos uno de los mediante **TCx_IRQn**, con x = 1, 2, 3, 4, 5, 6, 7, 8 o 9 representando uno de los nueve timers disponibles.

Una vez configurado el timer se puede inicializar o parar mediante los siguientes comandos:

TC_Start(TC1, 0); //Inicializamos el Timer1, canal 0
TC_Stop(TC1, 0); //Paramos el Timer1, canal 0

Con los pasos anteriores se ha realizado el procedimiento de configuración de un timer. El paso final sería definir la función que va a ejecutar dicho timer cuando se hayan completado los ciclos de reloj configurados. Para ello, fuera del set up ( ) y del loop ( ) se define la función, cuyo nombre hará referencia al timer.

```
void TC3_Handler()
{
  // Obtenemos el estado para volver a ejecutar el timer
  TC_GetStatus(TC1, 0);
}
```

El nombre de cada una de ellas será **TCx_Handler**, con x = 1, 2, 3, 4, 5, 6, 7, 8 o 9. Dentro de la función se debe obtener el estado del timer mediante **TC_GetStatus**, para volver a ejecutar el timer de manera ininterrumpida, a no ser que se pare.

Con todos los elementos anteriores, se tienen todas las herramientas necesarias para la elaboración de los algoritmos. En los siguientes apartados se detallarán cada uno de ellos, cuyo funcionamiento se basará en las configuraciones anteriores. La explicación de los mismos se llevará a cabo mediante diagramas de flujo para favorecer su comprensión. No se detallará el código en los siguientes apartados, si bien **en el CD que se adjunta como anexo al proyecto se pueden comprobar los sketches completos de cada algoritmo**.

4.1.2. ALGORITMO LINEAL BÁSICO

El algoritmo lineal básico se corresponde con la conversión analógica – digital de una señal empleando un contador lineal. En definitiva, se trata de sintetizar la parte digital de la figura 25, que junto con el integrado formen el conversor de doble rampa explicado en el capítulo 2, apartado 2.4.4. al que se le han realizado las modificaciones en el capítulo 3.

Para poder empezar con el desarrollo del algoritmo, se debe seleccionar la resolución que va a tener el CAD de doble rampa. Este aspecto es muy importante, ya que de él dependen parámetros como el periodo de reloj o el número de iteraciones que se van a realizar en el algoritmo. Para el conversor del presente proyecto, se va a seleccionar una **resolución de 8 bits**, lo que equivale a un total de 256 valores de conversión.

La **frecuencia del reloj** empleada en el incremento y decremento de los contadores se relaciona con la resolución ya citada. De forma general, se tiene que para una resolución de N bits, el periodo de reloj es inversamente proporcional a la enésima potencia N de dos de la resolución, o lo que es lo mismo:

$$Con\ N\ bits\ \rightarrow\ T_{clock} = \frac{1}{2^N}$$

Por lo tanto, para una resolución de 8 bits, el periodo del reloj valdrá T_{CLOCK} = 3.90μS ≈ 4μS.

Una vez se ha seleccionado la resolución del CAD de doble rampa, se debe tener en cuenta que la señal de entrada al mismo debe tener una resolución de dos bits adicionales a la del conversor. Es decir, si el CAD de doble rampa tiene una resolución de 2^N bits, la señal de entrada debe tener $2^N \cdot 2^2 = 2^{2+N}$. Esto se debe a que como señal de entrada se está empleando la salida analógica del CDA de Arduino, al cual se le debe configurar una resolución como se ha explicado en el apartado 4.1.1. Por lo tanto, como parámetros se van a tener:

- **Resolución del CAD de doble rampa**: 8 bits.
- **Periodo de relo**j: T_{CLOCK} = 4μS
- **Resolución del CDA de Arduino**: 10 bits.

Como se ha venido citando a lo largo de la memoria, un elemento clave en el desarrollo de los algoritmos va a ser el **timer**. Se va a emplear para controlar los tiempos T_1 y T_2 de la gráfica de la figura 27. El tiempo T_1 es un tiempo fijo, cuyo valor se puede configurar dependiendo de la aplicación. En él se va a realizar el decremento de un contador desde 2^N- 1 a 0. También podría realizarse el incremento de 0 a 2^N- 1, aunque para el caso que nos concierne se ha seleccionado la primera opción. Por lo tanto, el tiempo de conteo empleado en T_1 va a ser:

$$T_1 = T_{clock} \cdot 2^N$$

Dado que T_{CLOCK} = 4μS y 2^N- 1 = 256, el tiempo de duración de T_1 es 1mS. Se va a emplear un timer que se dispare una vez haya pasado un tiempo de 1mS, cuyo valor tendrá un número de ciclos de reloj determinado. Por lo tanto, el número de ciclos necesarios en el timer para obtener un tiempo de 1mS se obtiene con la siguiente ecuación:

$$nº.de\ ciclos = T_1 * TIMER\_CLOCK4$$

Se ha decidido emplear el reloj TIMER_CLOCK4 de 656,250KHz, aunque podría haberse empleado cualquiera de los timers disponibles. En consecuencia, el número de ciclos de reloj necesarios son 1mS · 656,250KHz = 656,56 ≈ 657 ciclos. Para implementar dicho timer se empleará la entrada de reloj externa TCLK3, cuyas características se muestran en la tabla 14.

Para el caso del tiempo T_2, la duración de este dependerá de la señal de entrada. Dependiendo de cuál sea el valor de la señal de entrada analógica, tendrá un tiempo u otro y finalizará cuando se produzca el cruce por cero, como se puede observar en la gráfica de la figura 27. En este tiempo es donde se produce el incremento del contador, cuyos incrementos se producirán cada 4μS. Por lo tanto, se debe implementar un reloj que cada 4μS realice un incremento y compruebe si se ha producido el cruce por cero. Para ello se empleará un timer que salte cada dicho tiempo. En este caso, el número de ciclos de reloj valdrá:

$$nº.de\ ciclos = T_{clock} * TIMER\_CLOCK2$$

Se ha decidido emplear el reloj TIMER_CLOCK2 de 10,5MHz, aunque podría haberse empleado cualquiera de los timers disponibles. Por lo tanto, el número de ciclos de reloj necesarios para disparar al timer es 4μS · 10,5MHz = 41 ciclos. Para implementar dicho timer se empleará la entrada de reloj externa TCLK6, cuyas características se muestran en la tabla 14.

Con los parámetros anteriores, se tienen todos los elementos necesarios para la elaboración del algoritmo. La idea será emplear dichos tiempos T_1 y T_2 e implementarlos en los timers.

En las funciones de los timers se realizarán el control de los pines del integrado y el incremento del contador.

De forma conceptual, la idea del algoritmo es como sigue: Primeramente se configuran los parámetros en el set up ( ), es decir, la resolución de los CAD de Arduino (**10 bits**), los **timers** con los valores anteriormente calculados y se seleccionan los pines del integrado para tener como entrada la señal analógica generada en el DAC0 de Arduino sin invertirla (**CLIRVin = 1** y **CLIpn = 1**). También se debe resetear el integrador por si está cargado el condensador y así tener condiciones iniciales nulas (**IMCLKR = 1**). Posteriormente se pondrá dicho pon a "0" para evitar que esté permanentemente a cero. Una vez se hayan configurado dichos parámetros, **se inicia el timer correspondiente a T_1,** y se deja parado el de T_2. Cuando se inicie T_1, se irá integrando la señal analógica obteniendo una recta con pendiente negativa. Una vez se hayan completado los 657 ciclos, equivalentes a 1mS, se para dicho timer, se conmuta de la señal analógica a la de referencia proporcionada por la salida DAC1 de Arduino (**CLIRVin = 0**) y se invierte para que tenga un valor negativo (**CLIpn = 0**). A partir de aquí **se inicia el timer correspondiente al tiempo T_2.** Se irá integrando la señal de referencia con valor negativa, por lo que se tendrá una recta con pendiente positiva. En paralelo se va disparando el timer cada 4μS, se incrementa el contador y se comprueba si se ha producido el cruce por cero. Una vez se produzca el cruce por cero, se para el timer y se obtiene el valor del contador el cual será proporcional al valor de la conversión. A partir de aquí se muestra el valor digitalizado en el monitor serie.

Dicha visualización se va a realizar en el loop ( ) una vez se haya producido el final de la conversión. Se empleará una variable booleana que en un primer momento se inicializará a "false", y se ponga a "true" cuando se produzca dicha conversión. En el procesado del loop ( ) se entrará únicamente cuando esté dicha variable a "true", de esta forma se liberan las tareas a realizar en los timers que pueden dar lugar a errores en la conversión.

El rango de valores de conversión de la señal de entrada analógica irá de 1,65V hasta 3,3V en el DAC0. Se parte de 1,65V ya que debido a que se está empleando la masa analógica, no puede haber valores por debajo de ella. Dichos valores se corresponde desde 512 hasta 1023 como parámetro de entrada al DAC0 de Arduino. También se debe elegir el valor de la tensión de referencia del DAC1, cuyo valor también irá de 1,65V a 3,3V, es decir, de 512 hasta 1023 como parámetro de entrada al DAC1. De forma experimental, se ha establecido que para un valor de referencia seleccionado en Arduino, la tensión de referencia en el conversor vale:

$$V_{REF\ DAC\ Arduino} = \frac{3.3V}{2^N} \cdot k; \quad con\ N = 10\ y\ 512 \leq k \leq 1023$$

$$V_{REF\ Conversor} = \left(V_{REF\ Arduino} - V_{REF\ Masa\ Analógica}\right) \cdot 2 =$$

$$(V_{REF\ Arduino} - 1.65V) \cdot 2$$

Esto es debido a que en el amplificador diferencial, además de acoplarse a cada entrada una tensión diferencial de Vd/2, se acopla una tensión de referencia de $V_{REF}/2$. En un

amplificador con una referencia de 0V, este segundo valor es nulo pero debido a la masa analógica es un parámetro a tenerlo en cuenta. En la siguiente figura se muestra la idea.

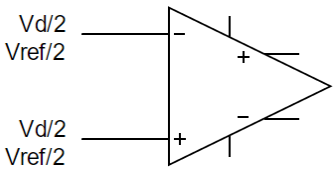


Figura 52: Acoplamiento de la señal diferencial y la señal de referencia a los terminales de entrada del amplificador diferencial

Por ejemplo, para una tensión de referencia de 1.95V en Arduino, la tensión de referencia en el conversor vale 0.6V.

Dado que la tensión analógica de entrada no puede superar a la de referencia, el algoritmo finalizará cuando la entrada sea igual a la analógica. En caso de superarse, la salida del conversor se saturará, dando lugar a resultados erróneos en la conversión. Este aspecto se podrá comprobar en el capítulo 5, apartado 5.2. Resultados preliminares del algoritmo lineal básico.

En el siguiente diagrama de flujo se resumen el funcionamiento del algoritmo, y en la tabla los parámetros de configuración de Arduino.

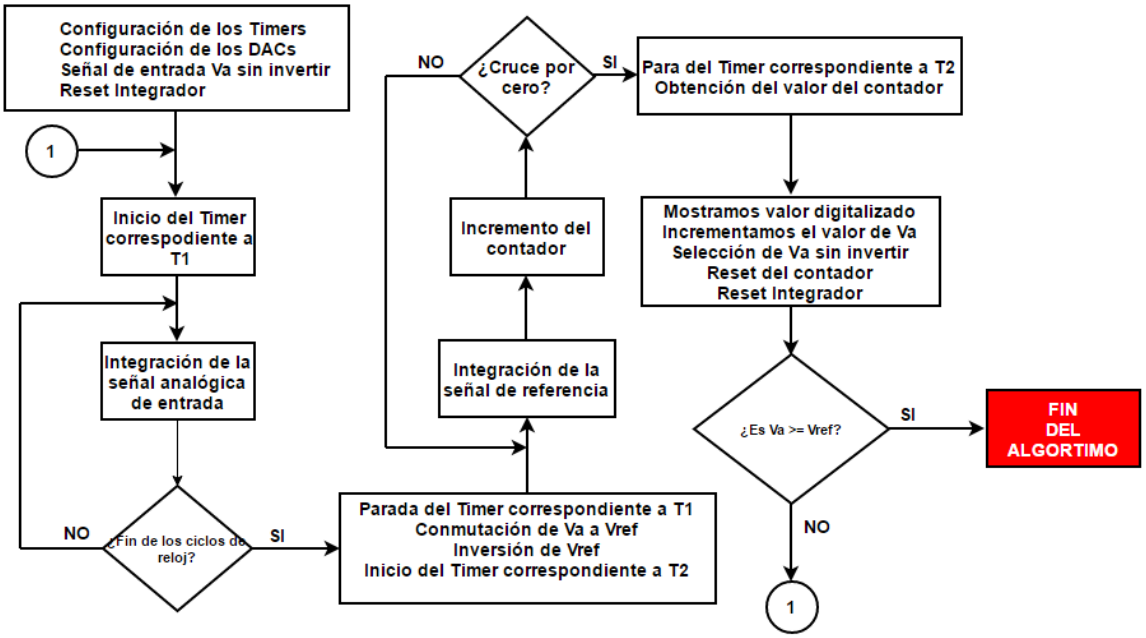


Figura 53: Diagrama de flujo del CAD lineal básico

Parámetros de configuración					
Resolución del CAD de doble rampa	8 bits				
Resolución del CDA de Arduino	10 bits				
Periodo del reloj	4μS				
TIMERS	**Entrada de reloj externa**	**Bloque**	**Canal**	**Nº. de ciclos**	**Tiempo equivalente**
Configuración del Timer correspondiente al tiempo T1	TCLK3	1	0	657	1mS
Configuración del Timer correspondiente al tiempo T2	TCLK6	2	0	41	Depende del periodo de integración de la señal

Tabla 15: Parámetros de configuración de Arduino

4.1.3. ALGORITMO NO LINEAL BÁSICO

Una vez se ha implementado el funcionamiento básico del CAD de doble rampa, se va a proceder al desarrollo del conversor analógico – digital junto con la linealización de la señal en la parte digital. En este algoritmo, la linealización se va realizar empleando una tabla o "array" que contenga los valores linealizados. De esta manera, una vez se ha cumplido el cruce por cero de una muestra e indique el fin de la conversión, se muestra el valor linealizado de dicha muestra. La idea es que dicho algoritmo sirva de puente entre el algoritmo de conversión lineal que emplea un contador lineal y el algoritmo de conversión no lineal que emplea un contador no lineal. De esta manera, se puede comprender su funcionamiento de una forma más intuitiva.

Como indica la propia palabra "linealizar", consiste en introducir una señal no lineal (p.e. un seno) y obtener a la salida una señal lineal (p.e. una línea recta). El objetivo de ello es integrar en un mismo sistema la parte de acondicionamiento y la parte de digitalización, simplificando el sistema de instrumentación en cuanto a complejidad y cantidad de componentes. Para comprender mejor el concepto de linealización, se muestra en la siguiente figura la idea.

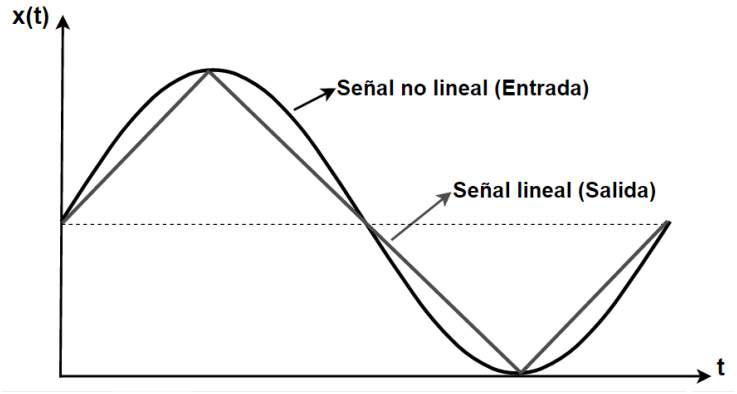


Figura 54: Proceso de linealización de una función no lineal

Como se puede observar, al introducir un seno, de carácter no lineal, obtenemos a la salida una señal linealizada. Aunque la figura anterior contemple todo el periodo de la señal sinusoidal, en el diseño del algoritmo se va a tener en cuenta únicamente **un cuarto de periodo de la señal.** El sentido de esto se tratará en el apartado 4.1.4., en el cual se explica la conversión no lineal con el contador no lineal. Por lo tanto, lo que se va a obtener con dicho algoritmo es lo siguiente.

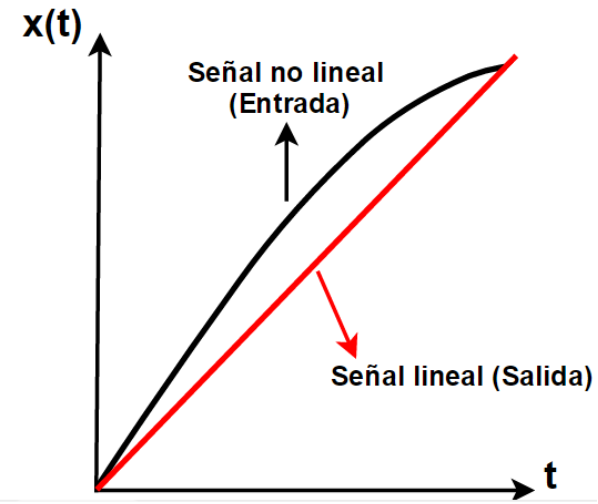


Figura 55: Proceso de linealización equivalente a un cuarto de periodo de la señal de entrada

Para la realización de este algoritmo, se va a emplear la configuración del apartado 4.1.2., cuyos parámetros se pueden consultar en la tabla X. Como se ha citado al principio de este apartado, para realizar la linealización se van a emplear dos tablas: una contiene los valores correspondientes al cuarto de periodo de la señal no lineal, que se van a introducir como parámetro de entrada al CDA de Arduino DAC0, y otra contiene los valores linealizados para cada muestra de la señal no lineal. En cada iteración del algoritmo, se irá incrementando el índice de la tabla correspondiente a la señal no lineal y se le aplicará el algoritmo de conversión. De esta manera, cuando se produzca el cruce por cero, se mostrará el valor linealizado de dicha muestra correspondiente con el mismo índice de la tabla que contiene los valores linealizados. Para este algoritmo no se va a emplear un contador que se incremente cada 4μS. Simplemente se detectará el cruce por cero, se mostrará el valor linealizado de la muestra y se pasará a la siguiente muestra correspondiente al siguiente índice de la tabla. Este proceso se repetirá hasta que se hayan digitalizado y linealizado cada una de las muestras. Para comprender mejor la idea de funcionamiento de las tablas, se muestra en la siguiente figura la idea.

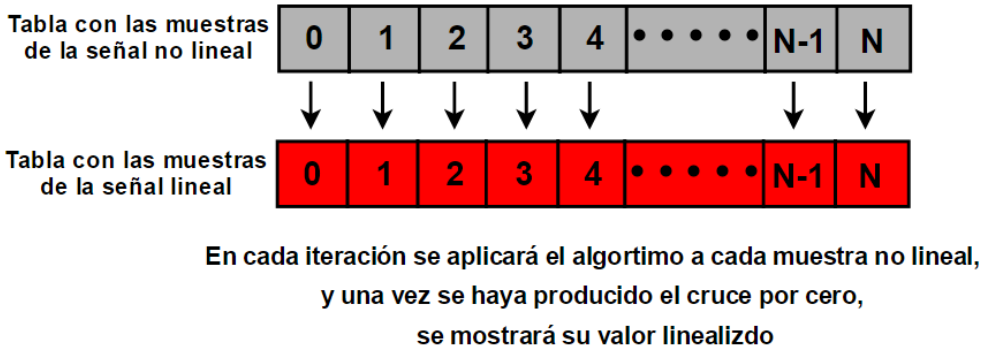


Figura 56: Correspondencia entre el valor no lineal y lineal en la tablas

La configuración de este algoritmo será idéntica a la del algoritmo básico lineal, salvo que se eliminará la idea del contador en el tiempo T_2. La definición de las tablas se realizará tanto dentro como fuera del set up ( ), si bien en el último caso deberá hacerse antes del set up ( ). El incremento del índice las tablas se realizará en el loop ( ) cada vez que se haya completado la conversión de una muestra. De nuevo, se emplearán los booleanos para mostrar los resultados una vez se haya completado la conversión. Finalmente se muestra la idea de funcionamiento del CAD de doble rampa no lineal básico a través del siguiente diagrama de flujo.

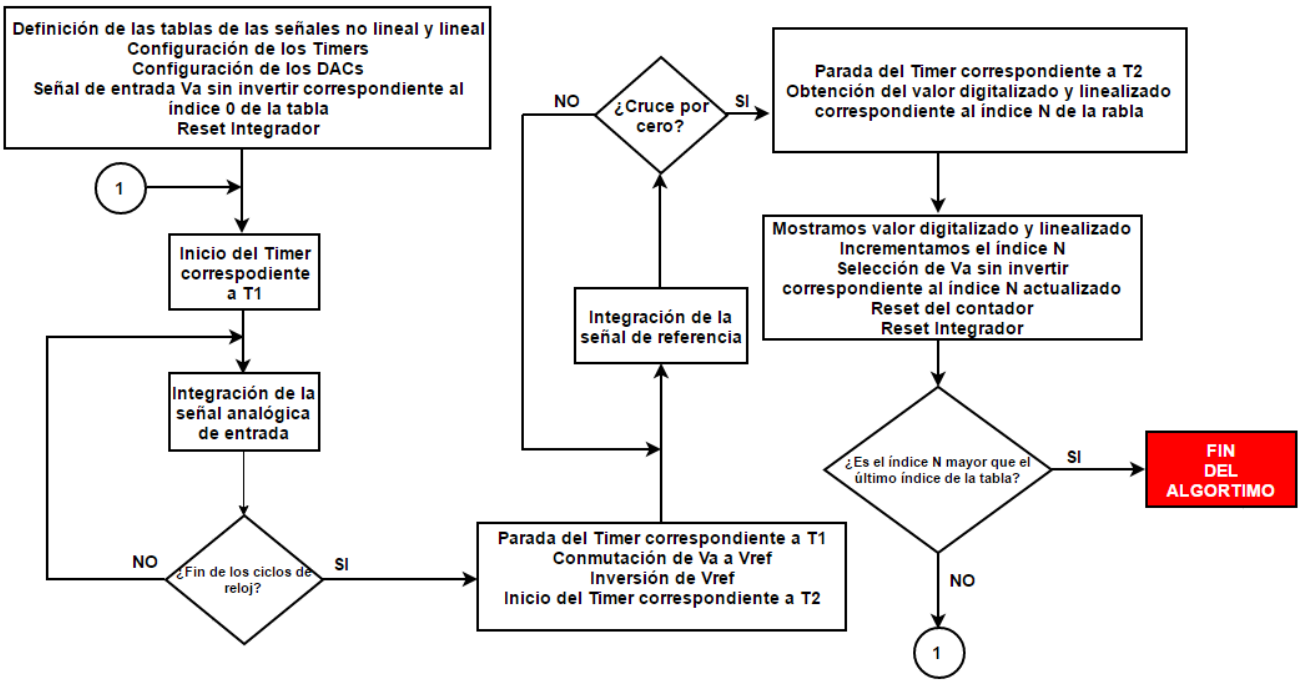


Figura 57: Diagrama de flujo del CAD no lineal básico

4.1.4. ALGORITMO NO LINEAL AVANZADO

Con el proceso de linealización explicado en el apartado anterior, se puede desarrollar el algoritmo de linealización basado en el **contador no lineal**.

Para poder conseguir dicho propósito, el CAD debe tener una función de transferencia inversa a la función de transferencia de la señal de entrada. Una manera de conseguirlo que permite una implementación sencilla consiste en linealizar por tramos la función inversa de la señal de entrada. Si la característica no lineal la aplicamos en el contador del CAD, la idea es que en cada ciclo de reloj el contador realice un incremento de acuerdo al tramo de aproximación lineal a tramos donde se encuentre el valor de entrada.

Dado que la señal de entrada va a ser un seno, la función inversa de este va a ser un **arco seno**. Por lo tanto, para linealizar por tramos dicha curva no lineal, el primer paso es encontrar los codos que delimitan cada tramo. Dos codos fijan la pendiente y constante de una recta de aproximación. Para ello se pueden emplear dos métodos:

- Dividir el rango X en intervalos iguales.
- Dividir el rango X de la función en intervalos según una función determinada, de manera que los intervalos más cortos estén en zonas donde la pendiente de la función varíe más rápido, haciendo que el error disminuya. Este es el método que se va a emplear para obtener los tramos de la función inversa, cuyo procedimiento fue elaborado en [1].

En la siguiente figura se muestra la función arco seno en un rango de 0 a 75º, dividida en 8 tramos. Cuanto mayor sea el número de divisiones, menor será el error cometido pero mayor será la carga computacional de Arduino. Por ese motivo se ha decidido emplear un valor que posea un error pequeño y una carga computacional no muy elevada.

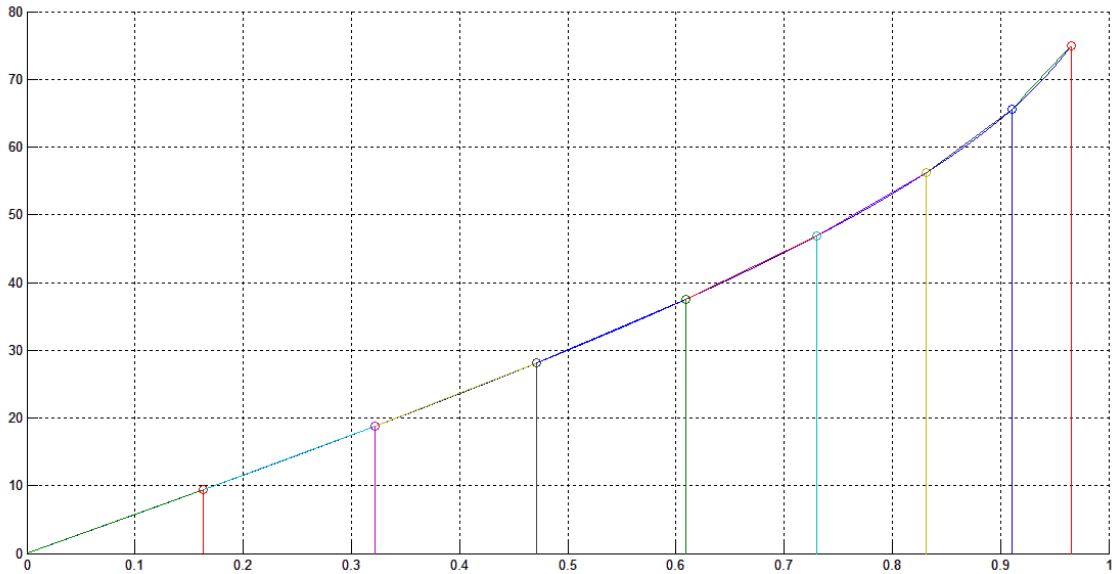


Figura 58: Linealización a tramos de la función arco seno

En el apartado anterior se ha citado que como señal de entrada se va a introducir un cuarto de periodo de un seno. Es debido a que la función arco seno está dividida en un rango de 0 a 75º y debido a ello no se puede coger todo el periodo del seno, sino un cuarto, o mejor dicho 75º.

En la siguiente tabla se muestra la posición de los codos y las características de cada pendiente de aproximación, correspondientes a la figura 58.

CODOS (X,Y)									
X	0	0,3478	0,5406	0,6767	0,7769	0,8506	0,9035	0,9414	0,9659
Y	0	20,3534	32,7256	42,5875	50,9775	58,2791	64,6283	80,2799	75
RECTAS y = mx + n									
n	0	-1,965	-6,448	-14,085	-25,97	-43,761	-70,446	-110,54	
m	58,519	64,169	72,461	83,746	99,044	119,96	149,492	192,8	

Tabla 16: Posición de los codos y las características de cada pendiente de aproximación

Con la función inversa de la señal de entrada dividida en sucesivos tramos, se va a proceder a comentar la idea del **contador no lineal**. Como se ha explicado, para el tiempo T_2 de la gráfica de la figura 27, en cada ciclo de reloj el contador aplica un incremento. El valor de este incremento, en vez de incrementarse en uno en cada ciclo de reloj, o sea, de manera lineal, va a realizar incrementos proporcionales al valor de la pendiente de las rectas de aproximación de la función inversa. Entonces el contador recibe como entrada la cuenta acumulada, la compara con los codos para conocer en qué recta de aproximación se encuentra y aplicarle así el incremento que le corresponda. Por ese motivo se le denomina contador no lineal, porque el incremento que se le aplica depende de en qué tramo de las rectas de aproximación se encuentre y cuyo valor será la pendiente de la recta de dicho tramo.

Dado que se está empleando un CAD de 8 bits, se debe adaptar la gráfica de la función inversa a dicha resolución. Para hacerlo se dividirá el eje X y el Y en 256 valores, y se re escalarán los valores de acuerdo a dicha resolución. En la siguiente figura se muestra la idea.

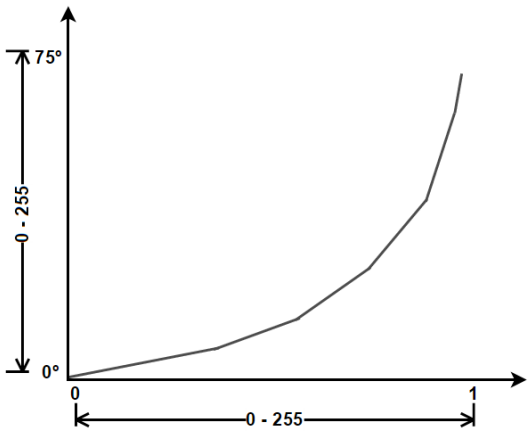


Figura 59: Re escalado de la función inversa a la resolución del conversor

En la siguiente tabla se muestran los valores re escalados de los ejes X e Y, así como el valor de la pendiente de cada tramo.

CODOS X									
X	0	0,3478	0,5406	0,6767	0,7769	0,8506	0,9035	0,9414	0,9659
X RE ESCALADA	0	92	142	178	205	223	237	247	255
CODOS Y									
Y	0	20,3534	32,7256	42,5875	50,9775	58,2791	64,6283	80,2799	75
Y RE ESCALADA	0	69	111	145	173	198	220	239	255
PENDIENTE DE CADA TRAMO $(Y - Y_O) / (X - X_O)$									
M	58,519	64,169	72,461	83,746	99,044	119,96	149,492	192,8	
M RE ESCALADA	0,75	0,84	0,94	1,04	1,39	1,57	1,9	2	

Tabla 17: Valores re escalados de los codos en X e Y así como de las pendientes de cada tramo

Con los valores anteriores se tienen todos los elementos para la elaboración del algoritmo. Un detalle importante antes de explicar el diagrama de flujo y la implementación del mismo es que cuando se obtenga el valor digitalizado, se le debe **sumar un valor de continua** que dependerá del tramo en el que haya caído la muestra. Es decir, supongamos que se le aplica el algoritmo a una muestra. Cuando se van sumando los valores de las sucesivas pendientes y acumulándose en el contador, únicamente se están teniendo en cuenta el valor de dichas pendientes en el cálculo del valor digitalizado, pero no los valores de continua de cada una de las rectas de aproximación. Por ese motivo, una vez digitalizado y linealizado el valor de las muestra, se le debe aplicar el valor de continua correspondiente al tramo donde haya caído. P.E. si ha caído en el tramo primero, se le sumará el nivel de continua de dicho tramo, que será cero. Para el segundo, se le sumará el del segundo, y así hasta completar los ocho tramos. En la siguiente gráfica se relaciona la curva inversa dividida en tramos, con las pendientes de cada tramo y los valores de continua de cada tramo.

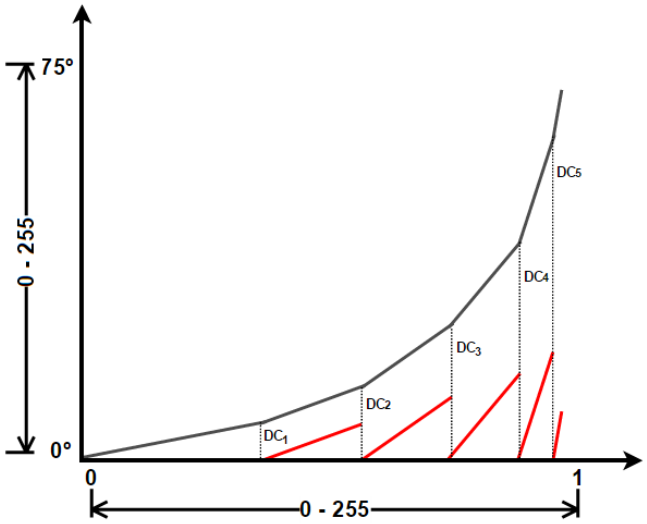


Figura 60: Nivel de continua a introducir en cada muestra una vez que se ha obtenido la cuenta

Con este último detalle, se va a proceder a explicar la implementación del algoritmo. Como en los dos anteriores algoritmos, los valores de configuración de los timers así como la resolución de los CDA de Arduino se pueden consultar en la tabla 15. En ese aspecto, no cambia nada con respecto a ellos. El tiempo T_1 será fijo, es decir, de 1mS y se llevará a cabo con su timer correspondiente. La diferencia va a erradicar en que para el tiempo T_2, el valor de incremento del contador no será constante, sino que dependerá del tramo en el que se encuentre el valor acumulado de la cuenta. Esto se realizará con el timer que corresponde al tiempo T_2. Para evitar sobrecargar al microcontrolador y hacer sumas con números decimales que incrementen el tiempo de procesado y puedan dar lugar a errores en la conversión, se van a emplear una serie de variables de tipo entero que marcarán el número de ciclos de reloj necesario para cada tramo. Es decir, una muestra caerá en el tramo primero si se han completado entre 0 y X_1 incrementos de reloj. Para el tramo segundo, entre X_1 y X_2 ciclos, y así sucesivamente. En la siguiente tabla se muestra el número de ciclos de reloj necesarios para cada tramo.

Ciclos de reloj necesarios para cada tramo	
Tramo 1	0 ≤ contador ≤ 92
Tramo 2	92 ≤ contador ≤ 142
Tramo 3	142 ≤ contador ≤ 178
Tramo 4	178 ≤ contador ≤ 205
Tramo 5	205 ≤ contador ≤ 223
Tramo 6	223 ≤ contador ≤ 237
Tramo 7	237 ≤ contador ≤ 247
Tramo 8	247 ≤ contador ≤ 255

Tabla 18: Ciclos de reloj necesarios para cada tramo

Aunque en la tabla se indique el valor del contador, en el algoritmo se emplearán variables de tipo entero para simplificar el procesado. El motivo de mostrar la variable contador en la tabla es por hacer una compresión más intuitiva. Una vez se produzca el cruce por cero y en consecuencia el fin de conversión, mediante las variables de tipo booleano anteriormente citadas entramos en la función del loop ( ). Ahí se calculará el valor digital linealizado sumando las variables de tipo entero acumuladas e incluyéndoles el valor de continua que le corresponda. Una vez mostrado el valor, se reiniciarán todos los contadores y se volverá a repetir todo el proceso.

Como ejemplo de señal de entrada se va a tomar un seno, aunque podría haberse empleado cualquier otra función no lineal. Los valores de dicha señal se incluirán en una tabla, la cual almacenará las muestras de la señal. Dichos valores serán los parámetros de entrada al CDA de Arduino. El algoritmo se aplicará para cada una de las muestras de la tabla. Se empleará un variable que vaya desplazando el índice de dicha tabla cuando se haya completado el proceso para la muestra anterior. Una vez se haya recorrido todos los índices, finalizará la conversión y en consecuencia el algoritmo. En la siguiente figura se muestra el diagrama de flujo que explica el funcionamiento del algoritmo.

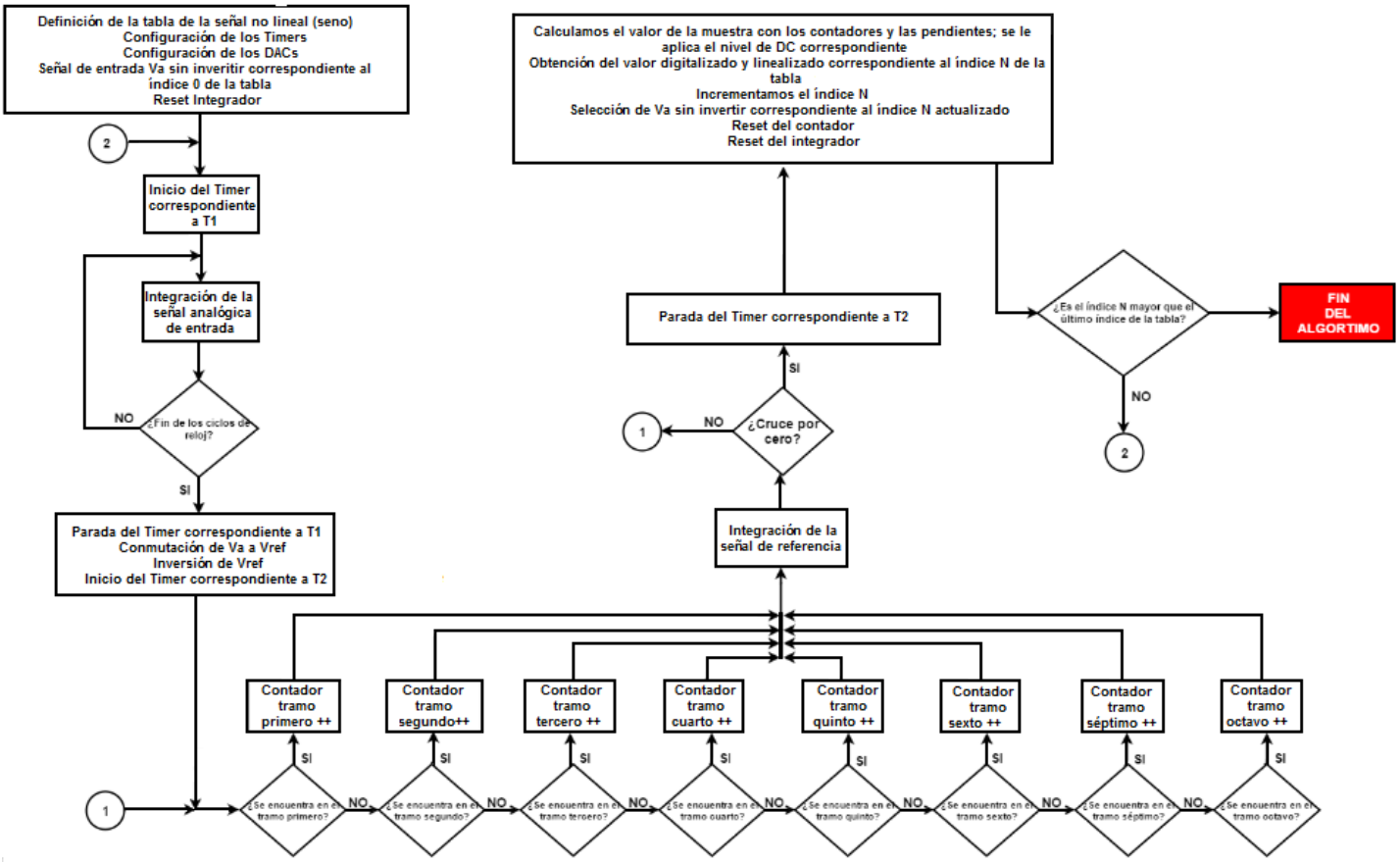


Figura 61: Diagrama de flujo del CAD no lineal avanzado

Con este último algoritmo, se ha completado el diseño del CAD no lineal. Primero se ha llevado a cabo el diseño del hardware mediante el diseño del shield. Posteriormente, se ha integrado con Arduino, formando la plataforma de implementación y testeo. El último comentario que habría que hacer son los resultados obtenidos experimentalmente de los diferentes algoritmos. En el siguiente capítulo se van a comentar los resultados obtenidos, desde el testeo de la placa hasta los algoritmos de conversión.

5. RESULTADOS

Una vez se tengan todos los elementos que forman plataforma, es decir, Arduino, el shield y los diferentes algoritmos de conversión, el paso final sería probarlos y extraer las medidas oportunas. En este capítulo se van a mostrar los resultados obtenidos de la plataforma. Primero se mostrarán algunas imágenes obtenidas mediante un osciloscopio para comprobar su funcionamiento. Seguidamente se mostrarán los **resultados preliminares** obtenidos en cada uno de los tres algoritmos implementados, en los cuales se incluirán varias gráficas que muestren los resultados.

5.1. TESTEO DEL INTEGRADO Y DEL SHIELD

Previamente a la implementación de los algoritmos, se debe comprobar que el hardware diseñado funciona correctamente. Para ello, una vez se haya elaborado y diseñado el shield, se debe integrar en Arduino. A continuación, se deben comprobar que todas las señales llegan a los puertos, la señal diferencial del integrador y finalmente la salida del comparador.

A lo largo de las sucesivas imágenes se van a mostrar varias medidas tomadas sobre la plataforma. En la figura 62 se comprueba la señal diferencial a la salida del integrador. Como se puede observar, aparecen dos señales: una verde y una amarilla. Dichas señales están medidas desde los terminales del integrado 31 a la tierra de Arduino (0V) y entre el 32 del integrado y la tierra de Arduino (0V). Dado que la salida es diferencial, se debe realizar la resta de ambas señales, es decir, la verde menos la amarilla, para obtener la señal diferencial deseada. Dicho resultado de hacer la diferencia es la señal violeta, que como se puede observar, es la señal triangular de la figura 27. Se pueden comprobar los dos tiempos T_1 y T_2, donde T_1 tiene una duración de 1mS y T_2 depende en este caso de la señal de entrada. Para el caso que nos concierne, T_2 tiene una duración de aproximadamente 400μS. Los pequeños picos que presenta la señal diferencial pueden deberse a ruidos de conmutación, ya que estamos trabajando con señales digitales cuyos valores son conmutados por los puertos de Arduino.

En la figura 63 se muestran las señales que controlan a los pines 4 (CLIpn) y 16(CLIRVin) del integrado. Se trata de señales digitales de valores alto – bajo, cuyos valores son o 0V o 3.3V. En ese caso, ambas señales conmutan en los mismos instantes para cambiar de la señal de entrada analógica a la de referencia e invertirla. Dicho cambio se realiza una vez terminado el tiempo T_1 y una vez se produce el cruce por cero en T_2, para volver al estado inicial del conversor y repetir todo el proceso. La señal verde controla al pin 4 y la amarilla al 16.

La señal de la figura 64 controla el reseteo del integrador. Cada vez que se produce un nivel alto, esta reinicia el integrador. Posteriormente, pasa a un estado de nivel bajo para no resetear el integrador de manera permanente. En los algoritmos, cuando se realiza el reseteo del integrador se espera un segundo a nivel alto y tras este tiempo vuelve al nivel bajo. El motivo ello es para que se produzca una descarga total del condensador.

Para comprobar que la salida del integrado proporciona el resultado de la comparación, se muestra en la figura 65 la señal a la salida de dicho comparador. Según el esquema de la figura 26, la salida del comparador estará a un nivel alto si no se ha producido el cruce por cero, ya que la tensión en la entrada no inversora es mayor que la de la inversora. En el momento que se produce el cruce por cero, la tensión en la inversora es superior a la de la inversora y la salida pasa a estar a nivel bajo. En ese tiempo a nivel bajo es donde se produce el procesado de la muestra, y donde se volverá al estado inicial para volver a aplicar el algoritmo a otra muestra.

Finalmente, la señal de la figura 66 muestra la tensión del pin 31 con respecto a la tierra de Arduino (0V) y la salida del comparador. No se ha podido mostrar la señal diferencial completa ya que no se disponía de más de dos canales para realizar la medida, pero dicha señal ilustra el concepto de una manera clara. Se comprueba por lo tanto que durante el tiempo T_1 y T_2 la salida del comparador está a nivel alto, ya que no se ha producido el cruce por cero. Una vez que se produce dicho cruce por cero, la salida del comparador para a nivel bajo. Esto indicará el fin de la conversión.

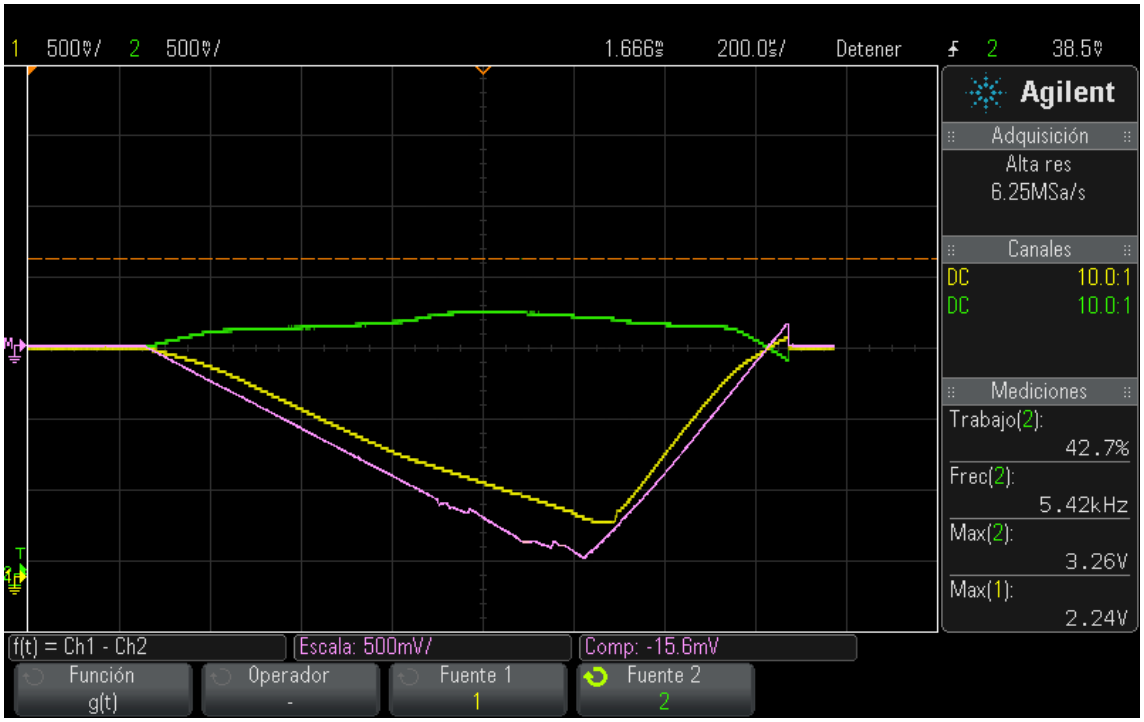


Figura 62: Señales a la salida del integrador diferencial con respecto a la tierra de Arduino así como la diferencial entre ambas

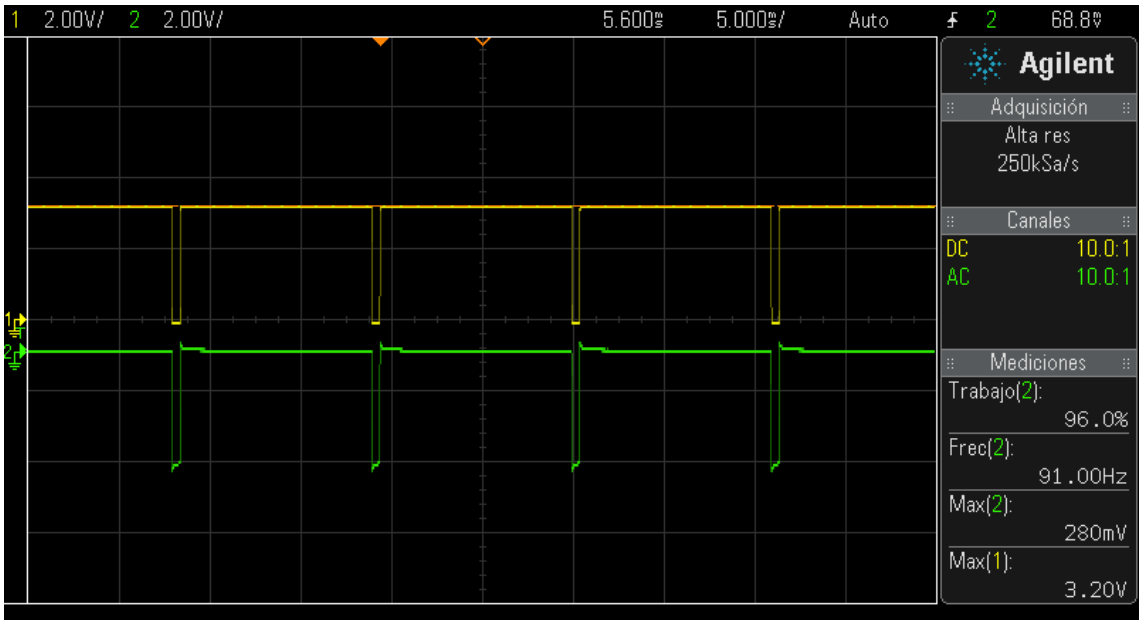


Figura 63: Pulsos digitales para el control de los pines 4 (CLIpn) y 16 (CLIRVin) del integrado

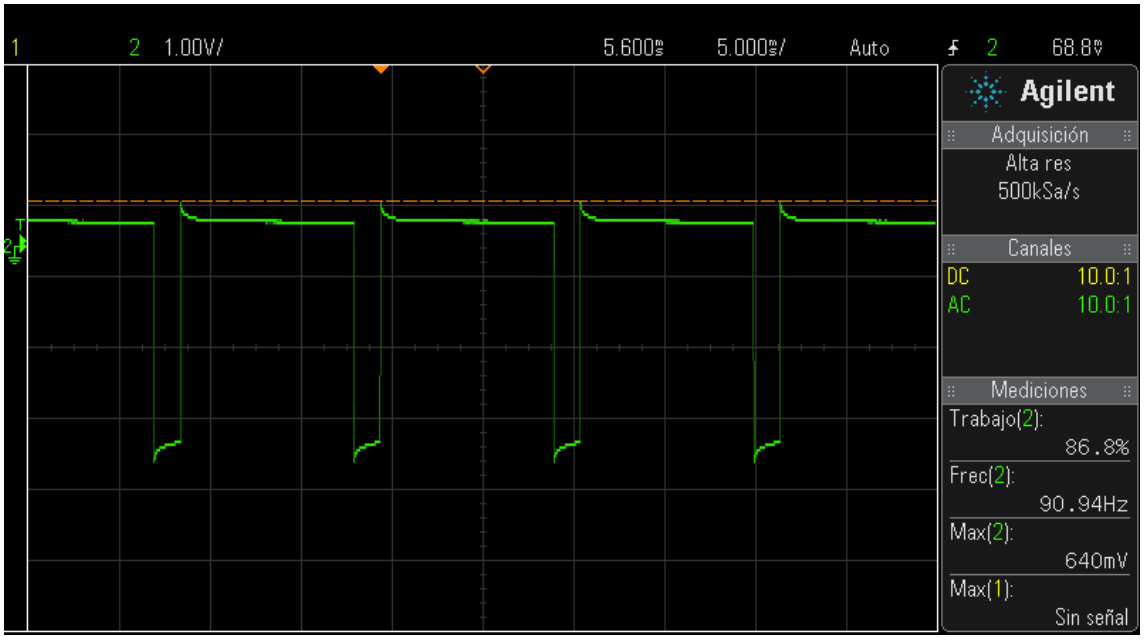


Figura 64: Pulsos que controlan el reseteo del integrador

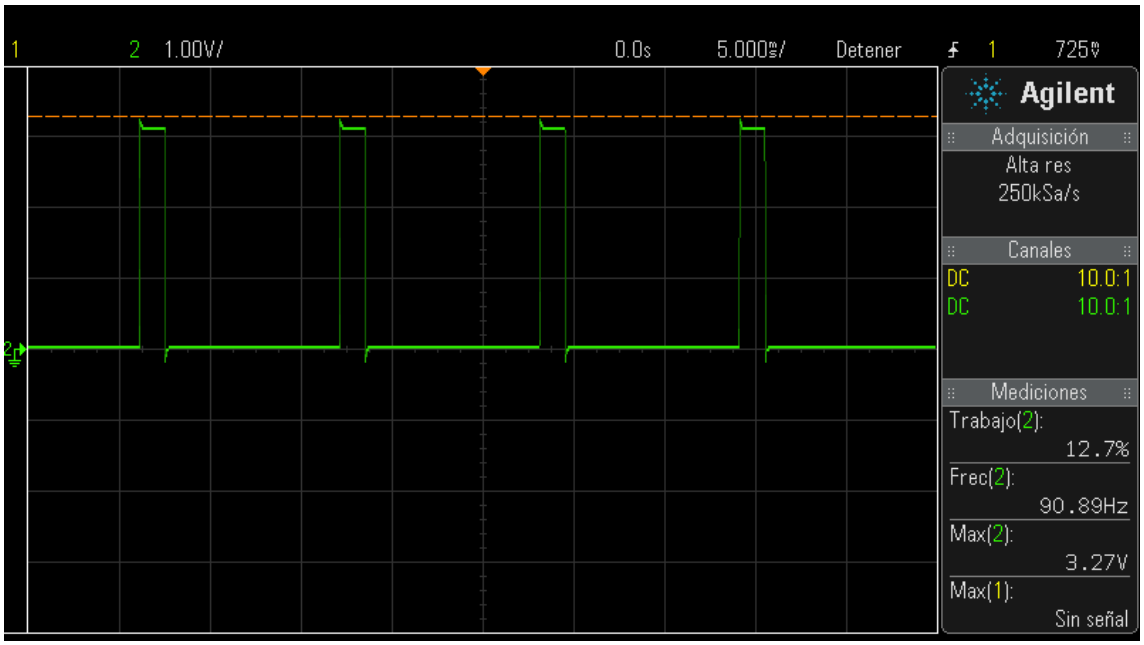


Figura 65: Señal que indica el fin de la conversión a la salida del comparador

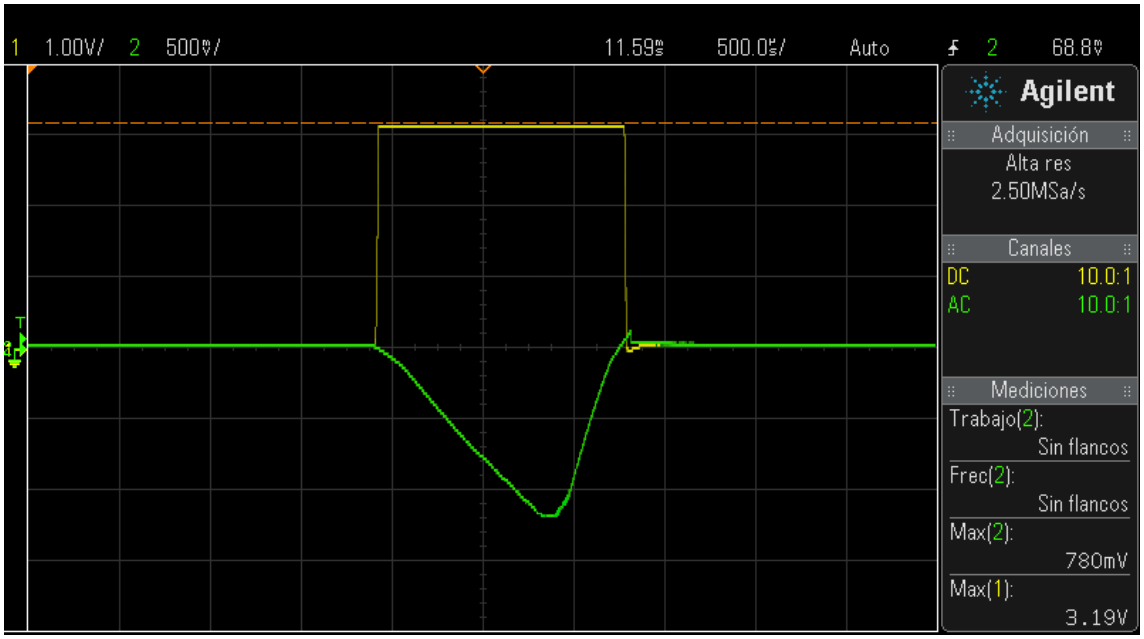


Figura 66: Señal en el pin 31 del integrado con respecto a la tierra de Arduino y la señal de comparación

Se ha decido no mostrar aspectos del testeo como las tensiones de alimentación debido a que mediante las figuras anteriores se demuestra el funcionamiento del shield, lo que implica que parámetros como las tensiones de alimentación van implícitos en dicho funcionamiento.

5.2. RESULTADOS PRELIMINARES DEL ALGORITMO LINEAL BÁSICO

Para poder testear el funcionamiento del algoritmo lineal básico se va a introducir como **entrada un conjunto de muestras proporcionadas por la salida DAC de Arduino**. Para ello se realizará un barrido desde 1.65V, lo que corresponde a 512 como parámetro de entrada al DAC, hasta 1.90V, que equivale a 590. Se seleccionará una **tensión de referencia** de 1.90V que equivale a un valor de 590. La elección de estos valores se debe a la explicación que se corresponde con la figura 52. La señal analógica de entrada nunca puede superar a la de referencia, por ese motivo cuando sea igual a esta finalizará el algoritmo.

El algoritmo irá procesando las muestras en las sucesivas iteraciones del mismo para obtener el valor digitalizado de dicha muestra. Una vez se haya aplicado el algoritmo sobre todas las muestras, finalizará todo el proceso de conversión. Como se ha citado, la conversión se terminará cuando la señal analógica de entrada sea superior a $V_{REF.}$

En la figura 67 se muestra la señal de entrada a introducir en el conversor. En el eje X se representa el valor de la muestra a introducir como parámetro de entrada a la salida DAC de Arduino y en el eje Y se muestra el valor de dicha muestra en voltios.

Así mismo, en la figura 68 se muestra el valor digitalizado para cada una de las muestras de entrada al conversor. La salida abarca prácticamente el rango de 0 a 255, siendo el valor mínimo 2 y el máximo 237. Una buena práctica para mejorar el rango sería modificar la pendiente de integración, lo que implicaría modificar el valor de los condensadores y resistencias del integrador. Dado que están fijos en el shield, no es posible su sustitución por otros valores, si bien dicha mejora se incluirá en el capítulo 6, apartado 6.2., futuras líneas. Otro aspecto a tener en cuenta es que la recta de conversión presenta alguna desviación. Puede ser debido a algún **error de linealidad** producido en el conversor o a los **efectos de la parte analógica** como el integrador. Deberían hacerse pruebas inyectándose más corriente en el pin 38 del integrado (IinBB), lo que implicaría modificar las resistencias de polarización de las fuentes de corriente externas.

Aunque no se muestre en la figura 68, si el valor de la tensión analógica superase al valor de la tensión de referencia, la gráfica sufriría una **saturación**. Este caso no es el deseado, por ese motivo nunca se debe superar a la tensión de referencia. Para ilustrar dicho concepto, se ha realizado un barrido mayor que el del caso anterior, en el que la tensión analógica va desde 1.65V o 512 como parámetro de entrada al CDA de Arduino hasta 2.89V o 899. Como tensión de referencia se han empleado 1.94V que equivale a un valor de 602 con el fin de maximizar el rango. En la figura 69 se muestra la señal de entrada, y en la figura 70 la señal digitalizada y saturada. Se puede comprobar que cuando la señal de entrada analógica supera a la de referencia, el sistema empieza a saturarse dando errores en la conversión.

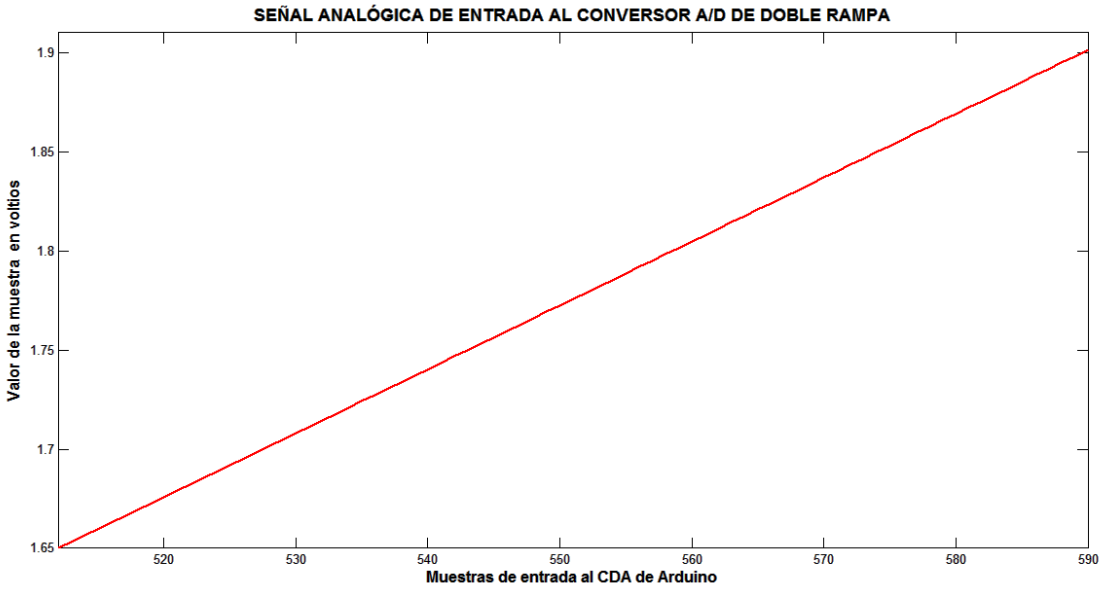


Figura 67: Señal de entrada para el CAD lineal básico

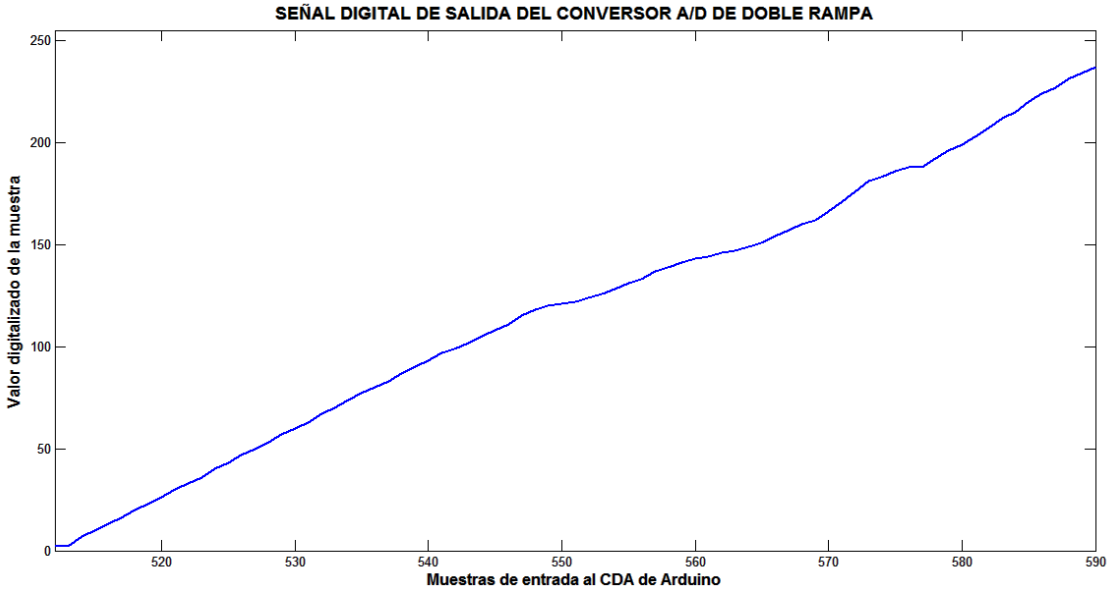


Figura 68: Señal digital a la salida del CAD lineal básico

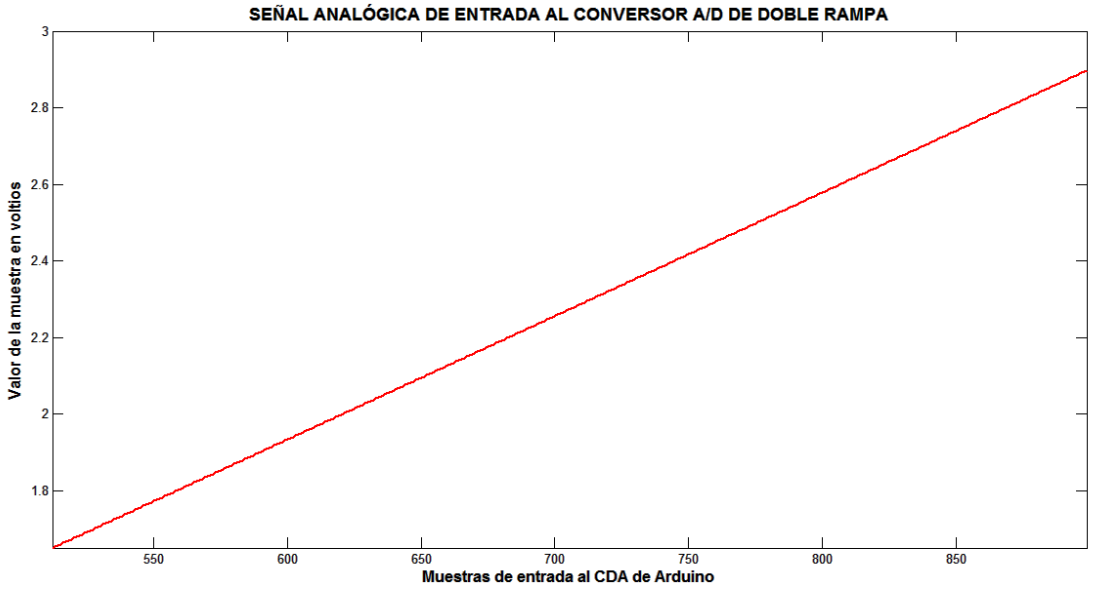


Figura 69: Señal de entrada para el CAD lineal básico con el fin de obtener saturación en la salida

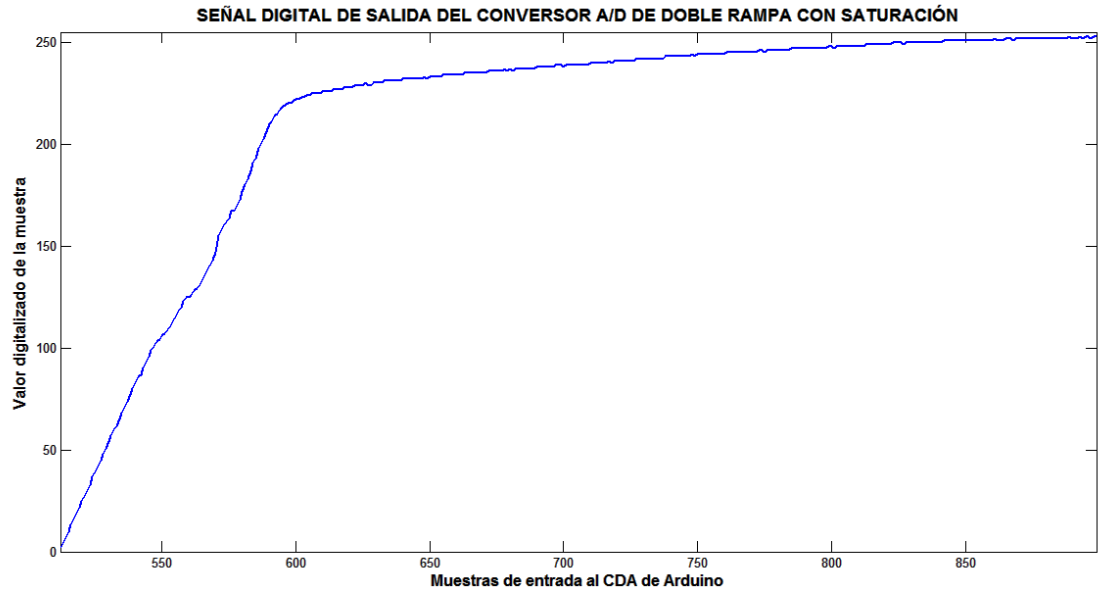


Figura 70: Señal digital a la salida del CAD lineal básico en el cual se ha producido saturación debido a que la tensión analógica de entrada ha superado a la de referencia

5.3. RESULTADOS PRELIMINARES DEL ALGORITMO NO LINEAL BÁSICO

El método de comprobación de este algoritmo es muy similar al del caso anterior. Como **señal de entrada al conversor se tienen 75º de la función seno**. Dicha función se obtendrá con 59 muestras que se corresponderán con cada uno de los puntos de la misma, cuyo rango irá desde 1.65V o 512 hasta 3.3V o 1023. Como **tensión de referencia** se va a emplear 3.3V o 1023 ya que el valor máximo de la función de entrada es 3.3V y nunca puede superar a la tensión de referencia.

Una vez tengamos todos los valores de las muestras, se introducirán en una tabla para que mediante un índice se vaya recorriendo y se le pase los valores al parámetro de entrada del DAC de Arduino. En otra tabla se tienen los valores linealizados de cada una de esas muestras de la función seno. Una vez se haya completado la conversión de una muestra, es decir se produzca el cruce por cero, se extrae el valor linealizado de dicha muestra. Dicho procedimiento se repetirá con todas las muestras de la función seno hasta finalizar.

En la figura 71 se muestra los 75º de la función seno que se va a introducir al CAD. En el eje X se representa el valor de la muestra a introducir como parámetro de entrada a la salida DAC de Arduino y en el eje Y se muestra el valor de dicha muestra en voltios.

Por último, en la figura 72 se muestra la salida del conversor. Se observa claramente que es la versión digitalizada y linealizada de la función seno.

Una de las ventajas que presenta este algoritmo es que se está empleando todo el **rango de conversión**. Además dado que se están empleando valores que están almacenados en una tabla, **nunca se producirán errores** en la conversión.

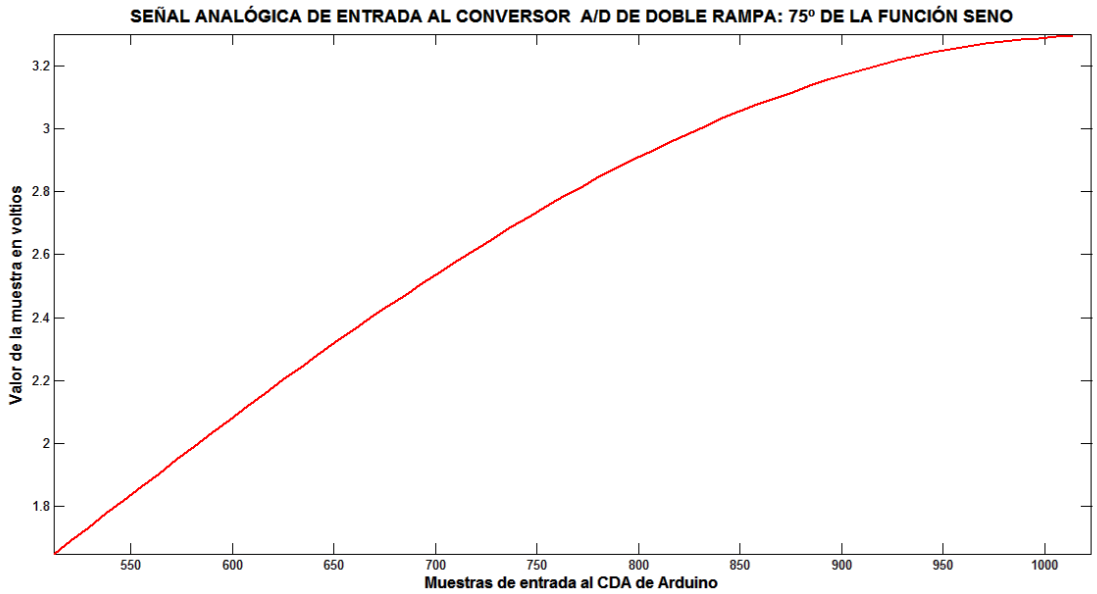


Figura 71: Señal de entrada al CAD no lineal básica, la cual se corresponde con 75º de la función seno

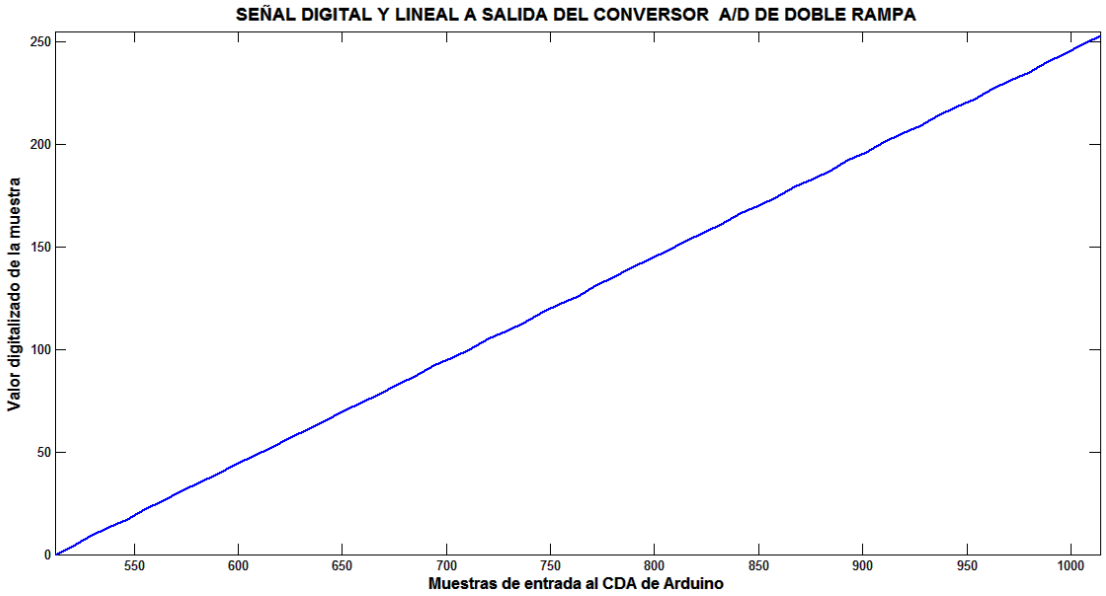


Figura 72: Señal de salida digital y lineal del CAD no lineal básico

5.4. RESULTADOS PRELIMINARES DEL ALGORITMO NO LINEAL AVANZADO

Para acabar con el testeo de los algoritmos, se va probar el algoritmo no lineal avanzado. Como ya se ha citado, este algoritmo se caracteriza porque **el valor de incremento del contador depende de la pendiente del tramo** en el que se encuentre el valor acumulado del mismo. Se irá **comparando con los codos** para saber en qué tramo se encuentra y aplicar el incremento en función de la pendiente de dicho tramo. Una vez se haya completado la conversión, se realizarán los cálculos oportunos así como la adición del nivel de continua para obtener el valor digitalizado y linealizado.

La **señal de entrada para este test es la misma que la del no lineal básico**, la cual se puede consultar en la figura 71. La tensión de referencia es de 1.94V o 602. Dado que el rango de la señal de entrada va entre 1.65V y 3.3V, en el momento que la muestra analógica supere a la de referencia se producirá **saturación** en el conversor. En la figura 73 se muestra la señal digitalizada y linealizada, en la que se produce saturación en el momento que la tensión de la señal analógica supera a la de entrada. La conversión es correcta en las siete primeras muestras, las cuales son menores que la tensión de referencia.

Una posible solución sería aumentar la tensión de referencia a 3.3V o 1023 pero en ese caso no se aprovecha todo el rango del conversor.

Por lo tanto, para obtener una conversión correcta se debe emplear una **señal de entrada comprendida entre 1.65V o 512 y la tensión de referencia**. En dicho rango se debe generar la señal no lineal de tipo senoidal. El problema es que es un rango muy pequeño para generar los 75º de la función seno. Al igual que en el conversor lineal básico, modificando los valores de las resistencias y condensadores que controlan la pendiente de integración, podría aumentarse el rango del conversor.

En la figura 74 se muestra el tramo de la salida el cual es lineal. Como se puede observar, abarca prácticamente todo el rango del conversor ya que la salida está comprendida entre 1 y 199. Esa pequeña desviación de la curva puede ser debida a algún **error de linealidad** producido en el conversor o a los **efectos de la parte analógica** del conversor como el integrador. Al igual que en el caso del conversor lineal básico, deberían hacerse pruebas inyectándose más corriente en el pin 38 del integrado (IinBB), lo que implicaría modificar las resistencias de polarización de las fuentes de corriente externas.

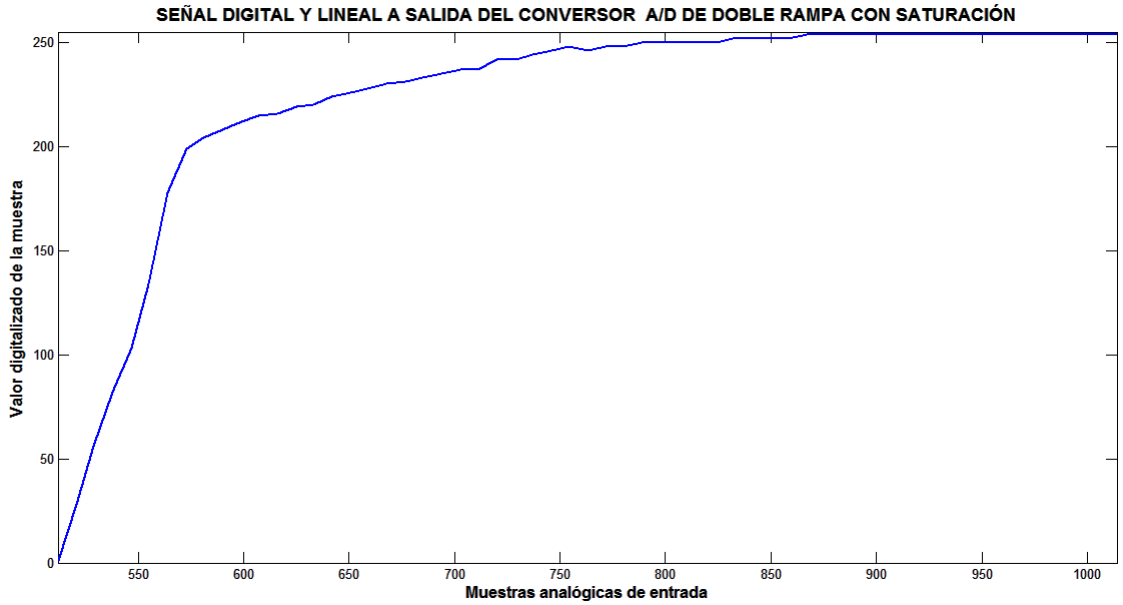


Figura 73: Señal digital y lineal a la salida del conversor; se puede observar cómo se satura el sistema en el momento que la señal de entrada analógica supera a la de referencia

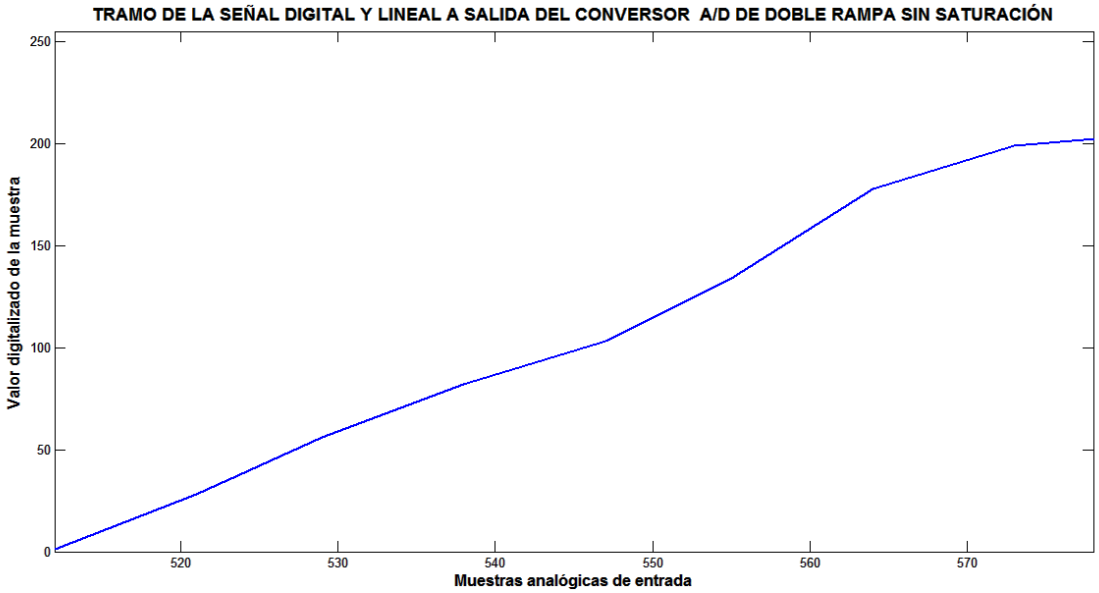


Figura 74: Tramo de la señal de salida sin saturación

6. CONCLUSIONES Y FUTURAS LÍNEAS

6.1. CONCLUSIONES

En el presente proyecto fin de grado se ha diseñado una plataforma de implementación y testeo para un **CAD no lineal de doble rampa mediante el sistema Arduino**. Se ha diseñado un PCB a modo de shield, el cual posee el integrado con la parte analógica del conversor, su circuitería asociada así como los pines que permiten incorporar el shield a Arduino. Mediante las diferentes entradas y salidas que posee Arduino, se controla al integrado y en consecuencia se realiza la conversión analógico – digital de la señal de entrada.

Una de las ventajas que presenta la plataforma es que es **altamente configurable**. Dado que el integrado es controlado íntegramente por Arduino, se pueden realizar múltiples algoritmos de conversión probando diferentes estrategias y estudiando diferentes escenarios. Además, dado que Arduino posee toda la circuitería necesaria para su funcionamiento, se reduce considerablemente el tiempo de diseño con respecto al uso de un microcontrolador habitual. En ese caso habría que haber diseñado, además del PCB con el integrado, todos los elementos necesarios para el funcionamiento del microcontrolador. De esta manera se evita la tarea de elaborar un PCB más complejo y en consecuencia reducir el tiempo de desarrollo.

Con respecto a los **algoritmos de conversión**, se han estudiado dos casos. Uno es el diseño de un CAD de doble rampa mediante el uso de un contador lineal cuyo objetivo es únicamente digitalizar una señal. El otro caso consiste en la implementación del CAD mediante un contador no lineal, que presenta un objetivo doble: digitalizar la señal y linealizarla. Este caso es útil en aquellas aplicaciones en las que se tenga una señal de entrada no lineal y se requiera su linealización. De esta manera, la linealización se consigue a través de la parte digital del conversor en vez de la analógica, lo que implica simplificar el diseño del integrado. También implica que el sistema de acondicionamiento sea más sencillo. Para el caso del conversor no lineal, se han elaborado dos algoritmos, uno basado en dos tablas, las cuales poseen la señal de entrada no lineal y la señal linealizada, y otro basado en el contador no lineal. El primero emplea un conteo lineal, y la linealización se consigue obteniendo los valores de esas tablas una vez finalizada la conversión. El segundo emplea el contador no lineal, en el que se realizan diferentes incrementos dependiendo de en qué rangos se encuentre el valor de la cuenta acumulada. Para obtener esos rangos, se debe coger la señal inversa de la señal de entrada, es decir, un arco seno en el caso de que la entrada sea un seno. Posteriormente se divide en varios tramos y la pendiente de cada tramo será el valor del contador no lineal. Cuantos más tramos posea, más preciso será el proceso de linealización, pero requerirá una carga computacional mayor.

Un aspecto a tener en cuenta en el diseño del algoritmo es que **la señal de entrada al conversor no puede superar a la tensión de referencia**, ya que en ese caso se satura la señal de salida. Es muy importante tenerlo en cuenta, ya que el algoritmo generará resultados erróneos.

El elemento clave de los algoritmos son los **timers**. Arduino presenta varios timers que pueden ser configurados y empleados de manera independiente. Su filosofía de funcionamiento es idéntica al de los timers de cualquier microcontrolador, pero presenta más características con respecto a un timer habitual dando lugar a más posibilidades.

Un aspecto muy importante en el desarrollo de los algoritmos es que el **tiempo** que emplea el microcontrolador en ejecutar una instrucción es un parámetro crítico para el proceso de conversión. En el caso del conversor lineal básico y del no lineal básico no supone un gran problema ya que las funciones que ejecutan los timers son muy sencillas, cuya única funcionalidad es la de conmutar los diferentes pines del integrado. Pero en el caso del conversor no lineal avanzado sí que es importante. Dado que se están realizando sumas y comparación de valores, el tiempo computacional será mayor y en consecuencia puede dar lugar a errores en el resultado de la conversión. Por ese motivo se debe optimizar las funciones, procurando emplear variables que impliquen menos carga como las de tipo entero en vez de tipo decimal. Una vez se haya ejecutado la conversión y obtenido el valor del contador, se paran todos los timers y se mostrarán los resultados de la digitalización. En el proceso de muestra de valores el tiempo ya no es un factor crítico, ya que el conversor no está digitalizando.

Otro tema que hay que remarcar es el empleo de la **masa analógica** del conversor. Su uso se debe a que Arduino trabaja con tensiones entre 0V y 3.3V y por lo tanto el conversor se debe alimentar con dichas tensiones. En consecuencia, el nivel de referencia del conversor será el valor medio de dichas tensiones, es decir, 1.65V. Este será un aspecto muy importante a la hora de testear el conversor, en el que las señales de entrada deberán estar comprendidas entre el valor de la masa analógica y el valor positivo de alimentación, es decir, entre 1.65V y 3.3V. No se debe confundir esta masa analógica con la masa de Arduino, las cuales son diferentes y sirven de referencia para cada sistema por separado.

6.2. FUTURAS LÍNEAS

Para mejorar la funcionalidad de la plataforma, podrían llevarse a cabo mejoraras tanto en el hardware como en el software de esta.

FUTURAS LÍNEAS DE DESARROLLO EN EL HARDWARE:

- Colocación de unos **jumpers** de conexión en las resistencias y condensadores que controlan la pendiente de integración así como en la resistencia de polarización de corriente. De esta manera, en vez de estar los componentes fijos en el PCB, podrían cambiarse dependiendo del valor de pendiente que deseemos.
- Otra alternativa a los jumpers sería emplear **potenciómetros** en las resistencias que controlen las pendientes de integración y la corriente de polarización.
- Diseño de un **circuito integrado** nuevo el cual incluya la parte digital desarrollada en Arduino. Dado que la señal de entrada es de carácter no lineal y existen múltiples tipos de señales no lineales, podrían incluirse en el diseño una serie de pines de control que elijan el tipo de entrada, y en función de esta tener su correspondiente función inversa. De esta manera, se tendría un circuito integrado de conversión analógico – digital no lineal para diferentes tipos de señales de entrada.
- Aunque el presente proyecto se ha centrado en el test de un CAD de doble rampa, la idea es crear un concepto de testeo que se pueda aplicar a cualquier sistema. Por ese motivo, se podría **estudiar otras arquitecturas de CAD** usando la filosofía de la plataforma propuesta en este proyecto fin de grado. Esto implicaría el diseño de un nuevo circuito integrado con la parte analógica del conversor, y mediante Arduino se generaría la parte digital de este.

FUTURAS LÍNEAS DE DESARROLLO EN EL SOFTWARE:

- Continuar trabajando para obtener el **ajuste adecuado** de las conversiones y obtener todo el rango y evitar saturaciones prematuras. Una calibración de las resistencias y condensadores o la base de tiempo de los contadores que controlan la pendiente de integración podría ser un buen comienzo. Una vez calibrado y mejorada la respuesta de la recta digitalizada y linealizada, se debería calibrar la tensión de referencia para encontrar el rango óptimo de funcionamiento.
- Para **evitar la saturación** del CAD, se podrían buscar nuevos algoritmos que realizaran la conversión en más de dos pasos de integración, intercalando los periodos descendiente y ascendente con el fin de evitar un voltaje máximo umbral. De esta manera los CAD diseñados podrían trabajar con fuentes de alimentación más bajas.
- Implementación de **nuevos algoritmos** de conversión no lineal. Estos podrían hacerse variando la frecuencia del reloj y haciendo un contador lineal. Es decir, en vez de realizar un conteo no lineal cuyo incremento dependa de la pendiente de la señal inversa, se llevaría a cabo un conteo lineal pero variando la frecuencia de reloj dependiendo de en qué tramo se encuentre. También podría realizarse el mismo procedimiento variando la referencia de tensión en función del tramo en que se encuentre, dejando fija la frecuencia de reloj y empleando un contador lineal.

- Desarrollo de una **pequeña interfaz** en la cual se introduzcan los parámetros del algoritmo y se envíen vía el puerto serie I_2C, en vez de modificarlo en el propio sketch.
- Elaboración de una macro en Excel o en cualquier otro lenguaje de programación como Procesing, la cual recoja los datos en tiempo real proporcionados por Arduino y los muestre en una **gráfica**.

7. BIBLIOGRAFÍA

[1] Bados Beraiz, M. (2007). Desarrollo de convertidores A/D de integración no lineales en tecnologías CMOS.

[2] Kester, W. (2005). The data conversión handbook. Newnes.

[3] Porta Cuellar, S. "Tema 3: Conversión". Procesado Digital de Señal. Universidad Pública de Navarra

[4] Karki, J. (2002). Fully – Differential Amplifiers. AAP Precision Analog: Texas Instruments.

[5] González de la Rosa, J.J. "Tema 11: Conversión Analógica/Digital". Instrumentación electrónica. Universidad de Cádiz.

[6] De La Cruz Blas, C.A. "Práctica 6: Introducción a OrCAD PCB editor". Universidad Pública de Navarra.

[7] Cuantificación digital. (2016, 2 de febrero). Wikipedia, La enciclopedia libre. https://es.wikipedia.org/w/index.php?title=Cuantificaci%C3%B3n_digital&oldid=88868597.

[8] Arduino Home. https://www.arduino.cc/

[9] Arquitectura ARM. (2016, 30 de abril). Wikipedia, La enciclopedia libre. https://es.wikipedia.org/w/index.php?title=Arquitectura_ARM&oldid=90776849.

[10] The Timer Counter blocks of Arduino Due's AT91SAM3X8E. https://github.com/ivanseidel/DueTimer/blob/master/TimerCounter.md

[11] Timer interrupts on Due. http://forum.arduino.cc/index.php?topic=130423.90

[12] SIK Experiment Guide for the Arduino 101/Genuino 101 Board. https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-the-arduino-101genuino-101-board/download-and-setup-the-arduino-software

[13 REF19xx Low – Drift, Low – Power, Dual – Output V_{REF} and $V_{REF}/2$ Voltage Reference, Datasheet. Texas Instruments.

[14] Atmel SAM3X / SAM3A Series, Datasheet. Atmel.

ANEXO 1: IMÁGENES DE LA PLATAFORMA

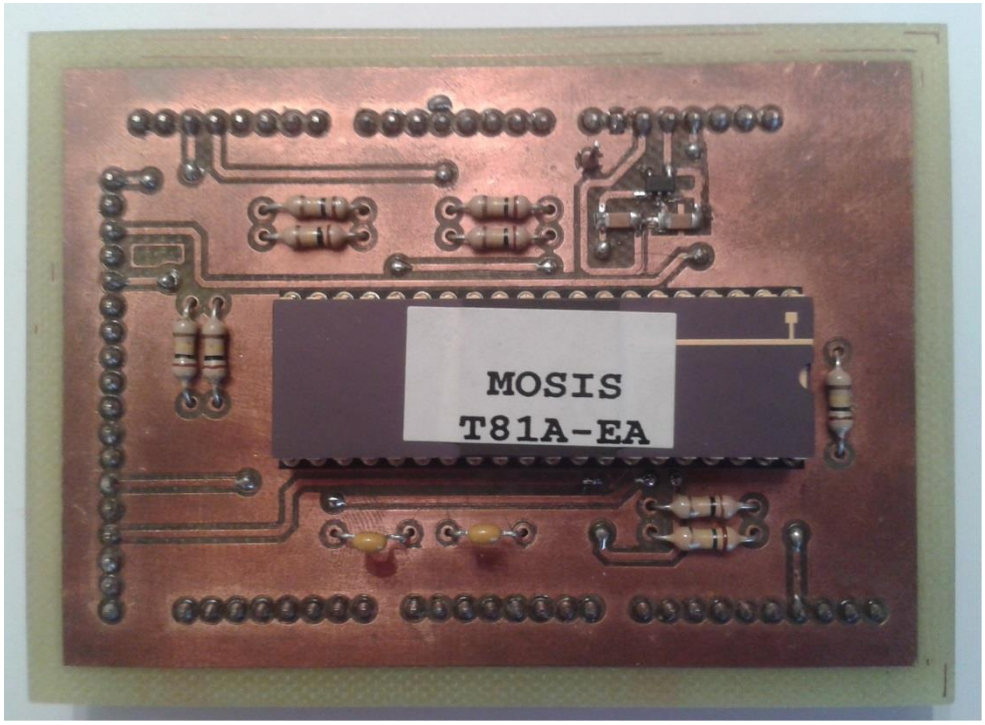


Figura 75: Imagen del shield con el integrado y su circuitería asociada

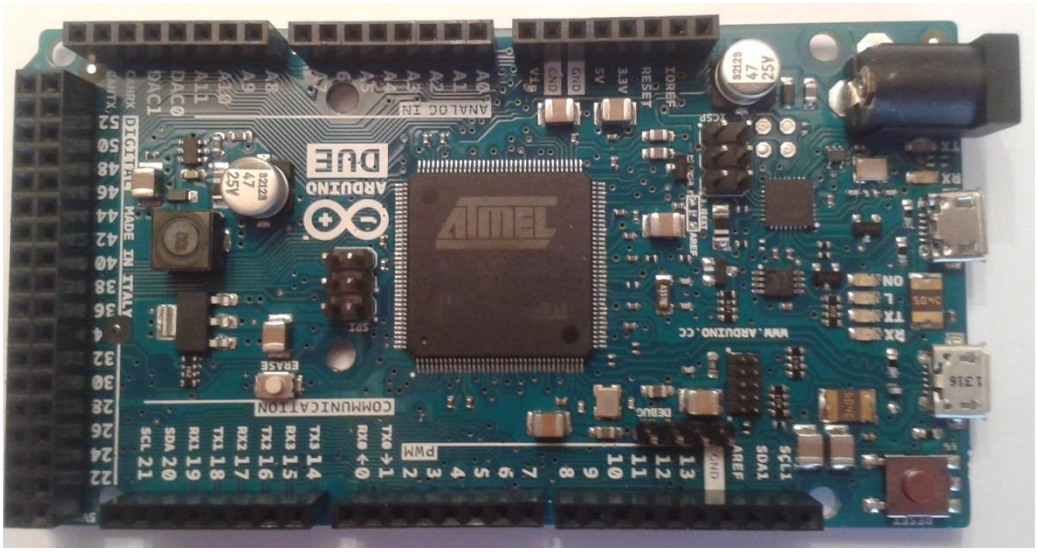


Figura 76: Imagen de la placa Arduino Due con el microcontrolador y su circuitería asociada

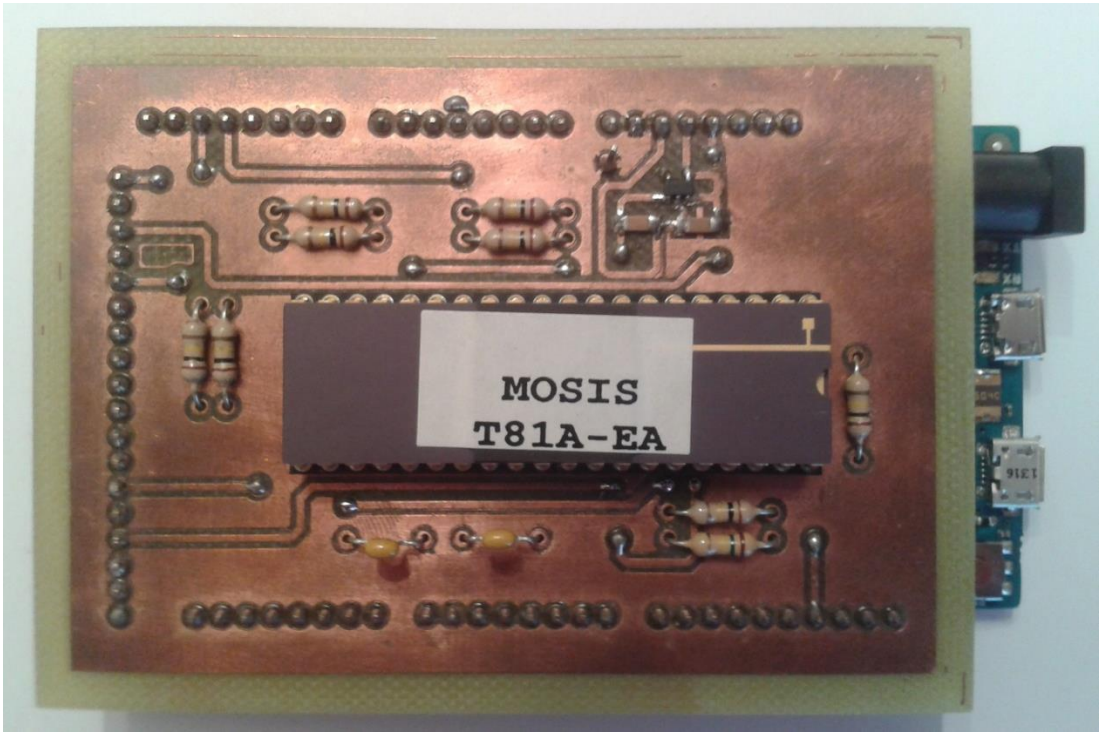
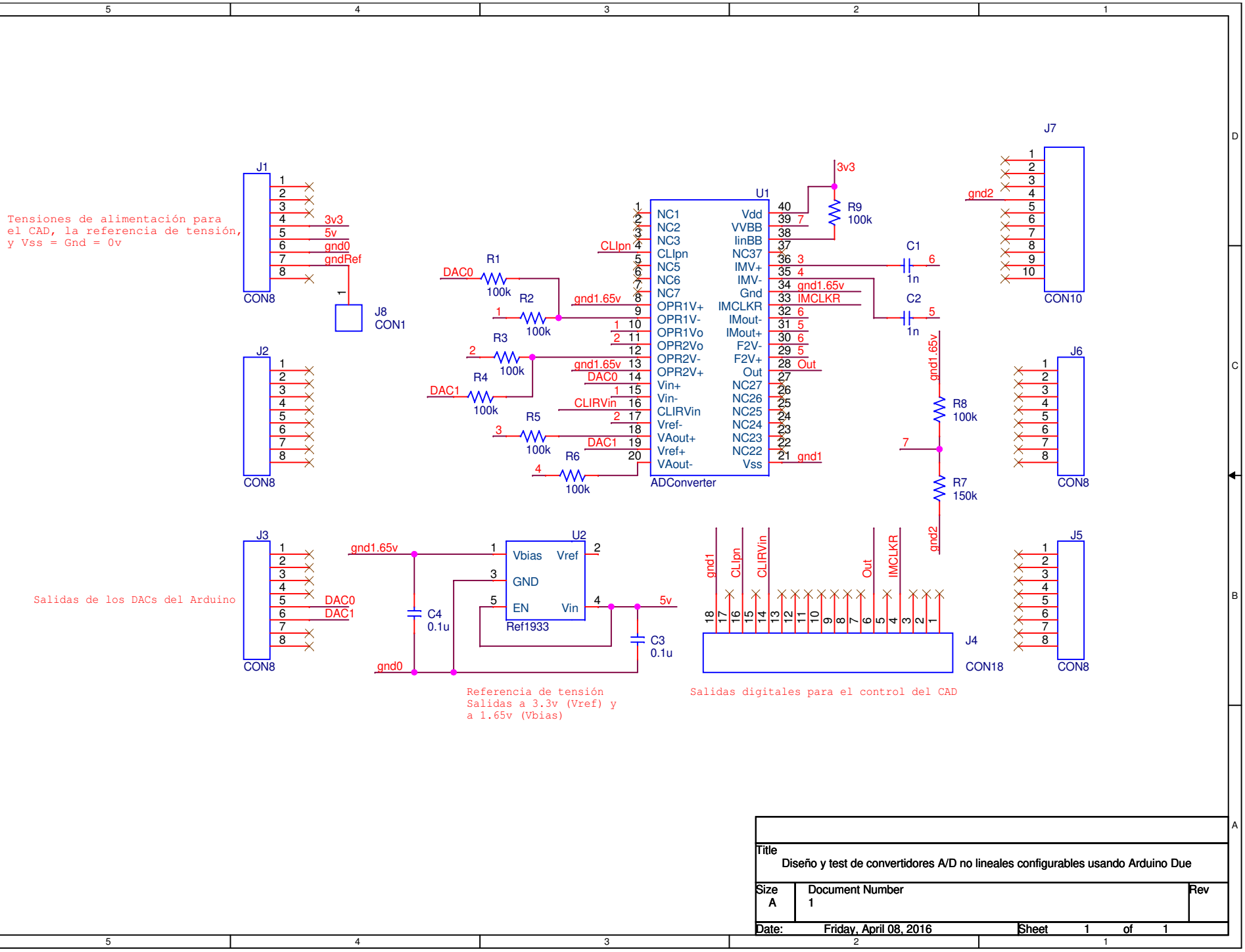


Figura 77: Imagen del shield montado sobre la placa Arduino Due

ANEXO 2: ESQUEMÁTICO DEL SHIELD


Tensiones de alimentación para el CAD, la referencia de tensión, y Vss = Gnd = 0v
J1
CON8
3v3
5v
gnd0
gndRef
J8
CON1
J2
CON8
J3
CON8
DAC0
DAC1
Salidas de los DACs del Arduino
U1
ADConverter
NC1
NC2
NC3
CLIpn
NC5
NC6
NC7
OPR1V+
OPR1V-
OPR1Vo
OPR2Vo
OPR2V-
OPR2V+
Vin+
Vin-
CLIRVin
Vref-
VAout+
Vref+
VAout-
Vdd
VVBB
IinBB
NC37
IMV+
IMV-
Gnd
IMCLKR
IMout-
IMout+
F2V-
F2V+
Out
NC27
NC26
NC25
NC24
NC23
NC22
Vss
gnd1
R1
R2
R3
R4
R5
R6
100k
R9
100k
C1
1n
C2
1n
R8
100k
R7
150k
3v3
gnd1.65v
gnd2
J7
CON10
J6
CON8
J5
CON8
J4
CON18
gnd1
CLIpn
CLIRVin
Out
IMCLKR
U2
Ref1933
Vbias
Vref
GND
EN
Vin
C4
0.1u
C3
0.1u
5v
gnd0
Referencia de tensión
Salidas a 3.3v (Vref) y
a 1.65v (Vbias)
Salidas digitales para el control del CAD
Title
Diseño y test de convertidores A/D no lineales configurables usando Arduino Due
Size
A
Document Number
1
Rev
Date:
Friday, April 08, 2016
Sheet
1
of
1